Chapter 15

Adaptive Interfaces and Agents

Anthony Jameson
DFKI, German Research Center for Artificial Intelligence /
International University in Germany
http://dfki.de/~jameson/

15.1 Introduction

As its title suggests, this chapter covers a broad range of interactive systems. But they all have one idea in common: that it can be worthwhile for a system to learn something about each individual user and adapt its behavior to them in some nontrivial way.

The perception and discussion of systems based on this idea has tended to be dominated by extreme examples, both real and hypothetical. So we should start with a look at a relevant system that is more typical of current trends (Figure 15.1). SWIFTFILE (Segal and Kephart, 2000) is designed to expedite the tedious task of filing incoming email messages into folders. By observing and analyzing the way an individual user files messages, the system learns to predict the three most likely folders for any new message. It enhances the usual email interface with three buttons that name the most likely folders. If the user notices that one of these guesses is correct, she can click on the corresponding button, saving herself the mental and physical effort of selecting the correct folder via the usual methods. If all of the guesses are wrong (a relatively rare event; cf. 15.7.2)—or if the user simply does not wish to pay any attention to the buttons—she can file the message in the usual way.

Concepts The key idea embodied in SWIFTFILE and the other systems discussed in this chapter is that of *adaptation to the individual user*. Depending on their function and form, systems that

Note: After copy-editing and acquisition of permission to reproduce figures created by other authors, this chapter will appear in the *Handbook of Human-Computer Interaction in Interactive Systems.*, edited by J. A. Jacko & A. Sears and published by Lawrence Erlbaum Publishers. The present preprint should not be posted on any web site.

EXHIBIT C

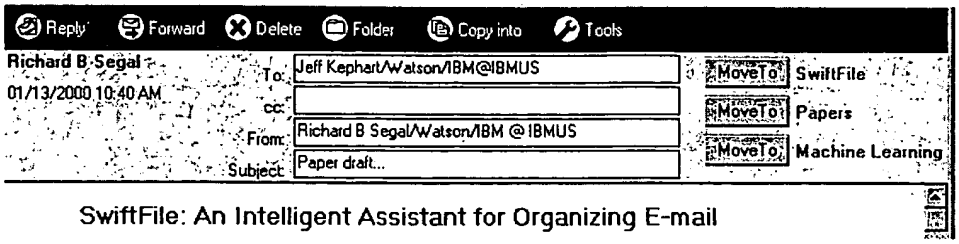

Figure 15.1. *Screen shot from* SWIFTFILE, *showing its three shortcut buttons for the filing of the current email message.*

(Part of Figure 1 of "Incremental learning in SwiftFile," by R. B. Segal and J. O. Kephart, 2000, in P. Langley (Ed.), Machine Learning: Proceedings of the 2000 International Conference, *San Francisco: Morgan Kaufmann. Copyright 2000 by Morgan Kaufmann Publishers.)*

adapt to their users have been given labels ranging from *adaptive interfaces* through *user modeling systems* to *software agents* or *intelligent agents*. Starting in the late 1990s, the broader term *personalization* became popular, especially in connection with commercially deployed systems. In order to be able to discuss the common issues that all of these systems raise, we will refer to them as *user-adaptive systems* ("UASs"). And to simplify exposition, we will use the symbol "$\mathcal{S}$" to refer to an interactive computing system or device and "$\mathcal{U}$" to refer to its user. Figure 15.2 introduces some concepts that can be applied to any UAS; Figure 15.3 shows the form that they take in SWIFTFILE.

A UAS makes use of some type of information about the current individual user $\mathcal{U}$, such as the choices $\mathcal{U}$ has made when filing messages into folders. In the process of *user model acquisition*, $\mathcal{S}$ performs some type of learning or and/or inference on the basis of the information about $\mathcal{U}$ in order to arrive at some sort of *user model*, which in general concerns only limited aspects of $\mathcal{U}$ (such as her mail-filing habits). In the process of *user model application*, $\mathcal{S}$ applies the user model to the relevant features of the current situation in order to determine how to adapt its behavior to $\mathcal{U}$.

A user-adaptive system can be defined as:

- An interactive system that adapts its behavior to individual users on the basis of processes of user model acquisition and application that involve some form of learning, inference, or decision making.

This definition distinguishes UASs from *adaptable* systems: ones which the individual user can explicitly tailor to her own preferences (for example, by choosing options that determine the appearance of the user interface). The relationship between adaptivity and adaptability will be discussed in 15.2.2 and 15.3.2.

Chapter Preview Sections 15.2 and 15.3 of this chapter address the question "What can user-adaptivity be good for?" They examine in turn nine different functions that can be served by user-adaptivity, giving examples ranging from familiar commercially deployed systems to research prototypes. Section 15.4 discusses some usability challenges that are especially important in connection with UASs, challenges which have stimulated most of the controversy that has surrounded these systems. The next two sections consider two key design decisions: What types of information about each user should be collected (Section 15.5), and what techniques should be used for the processes of learning, inference, and decision making that are involved in user model acquisition and application (Section 15.6)? Section 15.7 looks at several approaches to the empirical study

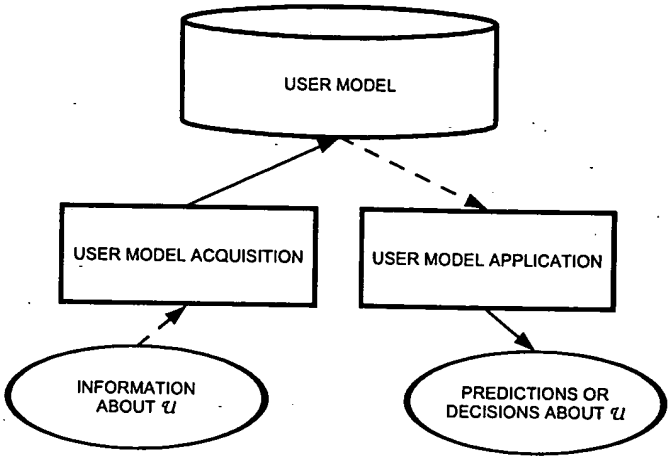

Figure 15.2. *General schema for the processing in a user-adaptive system.*
(Ovals: input or output; rectangles: processing methods; cylinder: stored information. Dotted arrows: use of information; solid arrows: production of results.)

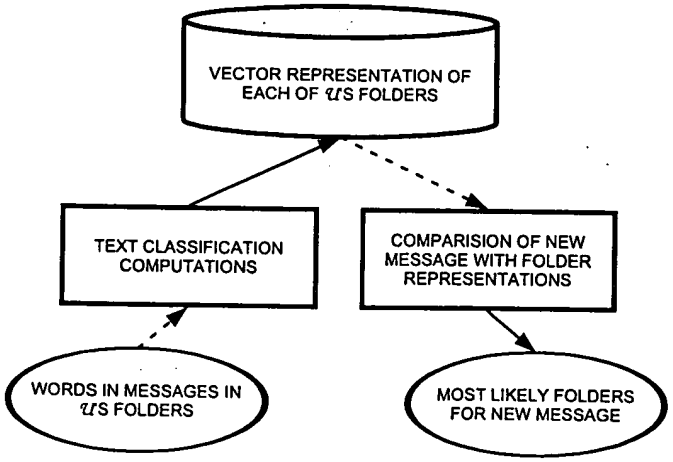

Figure 15.3. *Application of the schema of Figure 15.2 to the example of* SWIFTFILE.

of UASs, and the concluding section comments on the reasons why their importance is likely to continue to grow.

15.2 Functions: Supporting System Use

Some of the ways in which user-adaptivity can be helpful involve support for a user's efforts to operate a system successfully and effectively. This section considers four types of support.

15.2.1 Taking Over Parts of Routine Tasks

The first function of adaptation was illustrated by SWIFTFILE: Systems in this category take over some of the work that $\mathcal{U}$ would normally have to perform herself—routine tasks that may place heavy demands on a user's time, though typically not on her intelligence or knowledge. Typical tasks of this sort include email management (see, e.g., Maes, 1994) and appointment scheduling (see, e.g., Mitchell, Caruana, Freitag, McDermott, & Zabowski, 1994).
The primary benefits of this form of adaptation are savings of time and effort for $\mathcal{U}$. The potential benefits are greatest where $\mathcal{S}$ can perform the entire task without input from $\mathcal{U}$. In most cases, however, $\mathcal{U}$ is kept in the loop (as with SWIFTFILE), because $\mathcal{S}$'s ability to predict what $\mathcal{U}$ would want done is limited (cf. Section 15.4).

15.2.2 Adapting the Interface

A different way of helping a person to use a system more effectively is to adapt the user interface so that it fits better with $\mathcal{U}$'s way of working with the system. Interface elements that have been adapted in this way include menus, icons, and the system's processing of signals from input devices such as keyboards.

A recent example is provided by the SMART MENUS feature that Microsoft introduced in Windows 2000. Figure 15.4 illustrates the basic mechanism: An infrequently used menu option is initially hidden from view; it appears in the main part of a menu only after $\mathcal{U}$ has selected it for the first time. (It will be removed later if $\mathcal{U}$ does not select it often enough.) The idea is that in the long run the menus should contain just the items that $\mathcal{U}$ accesses regularly, so that $\mathcal{U}$ needs to spend less time searching within menus.

Although no published evaluation of SMART MENUS is available at the time of this writing, some relevant earlier research had been conducted prior to the appearance of SMART MENUS. For example, Sears and Shneiderman (1994) examined *split menus*, a different way of separating a menu into high- and low-frequency regions, with the several most frequently used options appearing at the top of the menu. In a field study and an experiment, split menus yielded generally positive results in terms of selection times and subjective preferences. These results confirm the potential benefits of this type of organization; but since the menus in these studies remained constant for each user, the results do not reflect the problems that can arise when menus change during use. Frequent changes in the arrangement of menu items can produce a strong variant of the general usability problem of *predictability*. As will be discussed in 15.4.1, this usability problem is especially acute for very frequently used interface elements, sometimes to the point of making virtually any form of adaptation undesirable.

One important goal of interface adaptation is to take into account special perceptual or physical impairments of individual users so as to allow them to use a system more efficiently, with minimal errors and frustration (cf. the chapters by Jacko et al., by Sears, and by Stephanidis in this handbook). Trewin and Pain (1998) have developed a method for recommending adjustments to

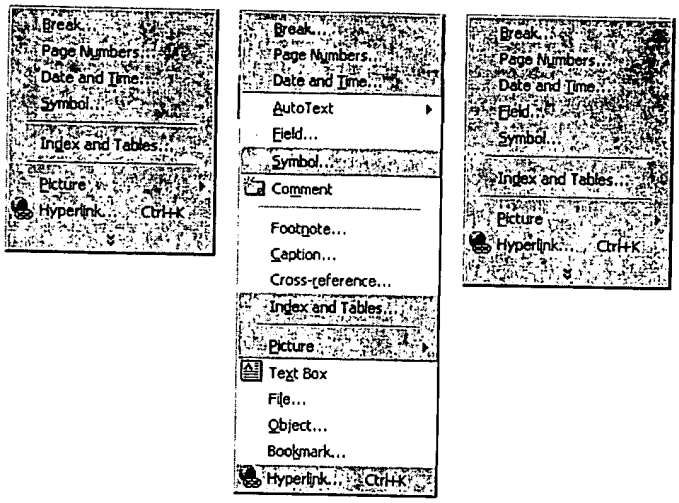

Figure 15.4. *Example of adaptation in* SMART MENUS.
(𝒰 accesses the "Insert" menu. Not finding the desired option, 𝒰 clicks on the extension arrows and selects the "Field" option. When 𝒰 later accesses the same menu, "Field" now appears in the main section.)

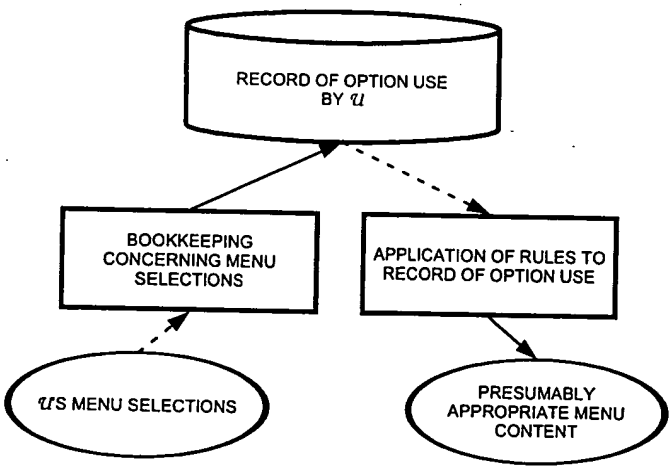

Figure 15.5. *Overview of adaptation in* SMART MENUS.

the parameters of a computer keyboard to compensate for several types of physical disability. For example, some users often inadvertently press the same key twice in succession. Trewin and Pain's system includes a mechanism for (a) recognizing this tendency on the basis of $\mathcal{U}$'s normal typing behavior and (b) computing an optimal "bounce key" interval, during which a given key cannot be reactivated. $\mathcal{U}$ is given the option of having the computed interval applied to her keyboard.

Many user interfaces, though not *adaptive*, are *adaptable* (cf. the discussion of this distinction in the chapter by Stephanidis in this handbook): They offer $\mathcal{U}$ the opportunity to specify desired properties of the user interface explicitly—for example, the aspects of the processing of keyboard input that Trewin and Pain (1998) address. Although *adaptability* is often an attractive alternative to adaptation, the keyboard example illustrates several typical limitations: $\mathcal{U}$ may not know what options exist or how she can set them. She may have no idea what the best setting is for her (e.g., the length of the optimal bounce-key interval), and trial and error with different settings can be time-consuming and frustrating.

One strong point of adaptability is that it leaves the user in control—a major usability consideration (Section 15.4). But as Trewin and Pain's recommender illustrates, it is often feasible to leave the final decision to $\mathcal{U}$ even when $\mathcal{S}$ assumes much of the burden of working out appropriate adaptations.

15.2.3 Giving Advice About System Use

Instead of suggesting (or executing) changes to the interface of a given application, a system can adaptively offer information and advice about how to use that application. As is discussed in the chapter by Mehlenbacher in this handbook, there exist various tendencies that make it increasingly difficult for users to attain the desired degree of mastery of the applications that they use. A good deal of research into the development of systems that can take the role of a knowledgeable helper was conducted in the 1980s, especially in connection with the complex operating system UNIX.[1] During the 1990s, such work became less frequent, perhaps partly because of a recognition of the fundamental difficulties involved. In particular, it is often difficult to recognize a user's goal when $\mathcal{U}$ is not performing actions that tend to lead toward that goal (cf. the results reported in 15.7.1).

The best-known adaptive help system is probably Microsoft's OFFICE ASSISTANT, which originally appeared in OFFICE 97. Part of the technology for the OFFICE ASSISTANT was derived from the LUMIÈRE prototype, which had been developed at Microsoft Research (see Horvitz, Breese, Heckerman, Hovel, & Rommelse, 1998). In Figure 15.6, LUMIÈRE is proposing help topics on the basis of $\mathcal{U}$'s recent actions—a source of information that can be especially valuable when $\mathcal{U}$ requires some advice but does not know which concepts she could use to find it in either the system itself or a nonadaptive help system. The type of spontaneous intervention shown in Figure 15.6 can also help $\mathcal{U}$ to expand her knowledge of the system's functionality, even when she is able to perform her tasks with the functions that she is already familiar with.

When adaptive help is given spontaneously, and not just on demand, it can endanger the usability goals of *unobtrusiveness* and *controllability* (see 15.4). Figure 15.6 illustrates one way in which the original LUMIÈRE prototype dealt with this problem, using decision-theoretic methods (15.6.3): The assistance window in Figure 15.6 appears only if the "Likelihood that help is needed" by $\mathcal{U}$ exceeds a given threshold. The window is initially small, and it includes a "volume control" with which $\mathcal{U}$ can raise or lower the threshold for $\mathcal{S}$'s future interventions. If $\mathcal{U}$ does not hover over the assistance window or interact with it, the window spontaneously vanishes after displaying a brief apology in its title bar.

[1] A collection of papers from this period recently appeared in a volume edited by Hegner, McKevitt, Norvig, and Wilensky (2001).

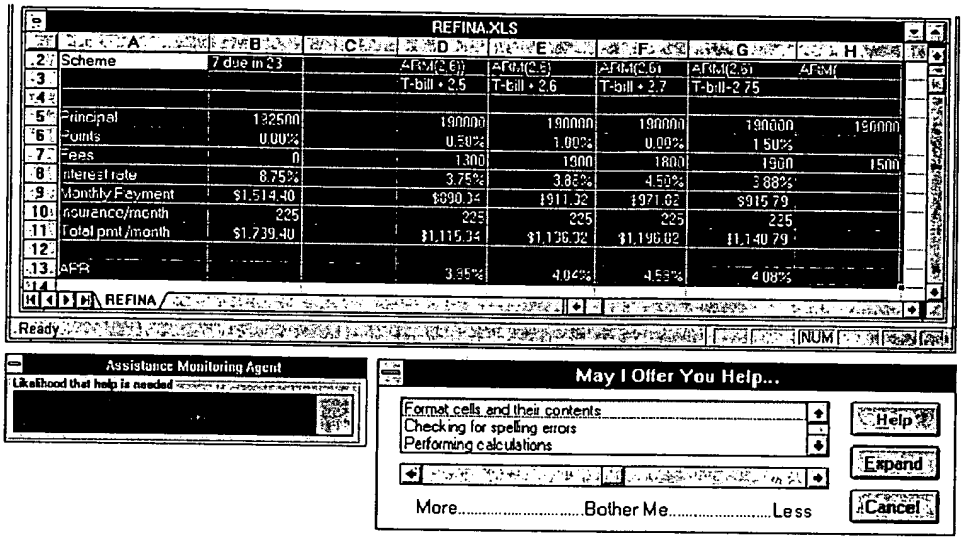

Figure 15.6. *Example of assistance offered by the* LUMIÈRE *prototype.*

(U has just searched through several menus, selected the entire spreadsheet, and paused. Figure 9 of "The Lumière project: Bayesian user modeling for inferring the goals and needs of software users," by E. Horvitz, J. Breese, D. Heckerman, D. Hovel, and K. Rommelse, 1998, in G. F. Cooper & S. Moral (Eds.), Uncertainty in Artificial Intelligence: Proceedings of the Fourteenth Conference, *pp. 256–265, San Francisco: Morgan Kaufmann. Copyright 1998 by Morgan Kaufmann Publishers.)*

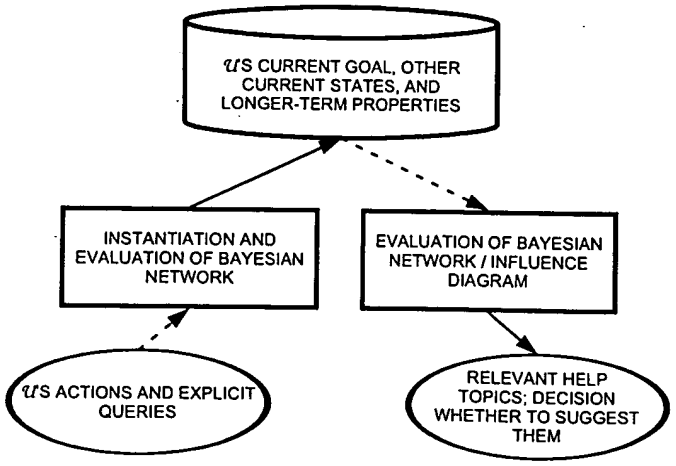

Figure 15.7. *Overview of adaptation in* LUMIÈRE.

In the deployed OFFICE ASSISTANT, the decision-theoretic methods for deciding when to offer help or were replaced with a relatively simple rule-based system which, for example, did not take into account $\mathcal{U}$'s level of competence or her willingness to be interrupted. The resulting tendency of the OFFICE ASSISTANT to pop up in "distracting" ways (see, e.g., Schaumburg, 2001) has created a rather distorted impression the potential value of adaptive help systems. More recent research has continued to explore ways in which UASs can take into account users' cognitive resource limitations with decision-theoretic methods (see, e.g., Horvitz, 1999; Horvitz, Jacobs, & Hovel, 1999; Jameson et al., 2001; Wolfman, Lau, Domingos, & Weld, 2001).

15.2.4 Controlling a Dialog

Much of the early research on UASs concerned systems that conducted natural language dialogs with their users (see, e.g., Kobsa & Wahlster, 1989). During the 1990s, attention shifted to interaction modalities that were more widely available and that made it possible in many cases to implement adaptation straightforwardly. Advances in the technology of natural language and speech processing (cf. the chapters by Karat et al. and by Lai & Yankelovich in this handbook) have led to a recent reawakening of interest in user-adaptive dialog systems (see, e.g., Haller, McRoy, & Kobsa, 1999; Zukerman & Litman, 2001).

Natural language dialog has served as an interaction modality in connection with most of the functions of user-adaptivity discussed in this and the following sections, such as the provision of help and the recommendation of products. But there is also a type of adaptivity which is largely characteristic of natural language dialog: adaptation of the system's *dialog strategy*: a policy for determining when and how $\mathcal{S}$ should provide information, acquire information from $\mathcal{U}$, and perform other dialog acts.

An informative example is TOOT (Litman & Pan, 2000), a prototype spoken dialog system for retrieving online train schedules. The basic idea underlying the adaptation is that different dialog strategies are appropriate in different situations. For example, if a given $\mathcal{U}$'s speech is understood poorly by $\mathcal{S}$, $\mathcal{S}$ may need to adopt a conservative dialog strategy, acquiring just one piece of information at a time and frequently requesting confirmation. If, on the other hand, $\mathcal{U}$'s speech is understood well by $\mathcal{S}$, more efficient dialog strategies can be used. Figure 15.8 shows two excerpts from an example dialog in which $\mathcal{S}$ starts with an ambitious, potentially efficient strategy and then, noticing that the dialog is not progressing well, shifts downward twice until it finally achieves success with a very conservative strategy.

In a controlled empirical evaluation of TOOT's adaptation (described by Litman & Pan, 2000, and discussed below in 15.7.3), it was shown that TOOT adapted to different extents in different dialogs and that this adaptation led to a much higher rate of successful dialog completion (65% vs. 23%) than the use of a single dialog strategy (the first one shown in Figure 15.8). User-adaptivity also led to a slight increase in subjective satisfaction. Some issues that arise in the design and interpretation of controlled studies of this type will be discussed in 15.7.3.

15.3 Functions: Supporting Information Acquisition

We are constantly hearing that information overload is a typical problem of our age, especially because of the explosive growth of the internet and in particular the world-wide web. In addition to the vast number of electronic documents of various sorts, users now have access to a vast number of products available for sale, people that they can get in touch with, and systems that can teach them about some topic. The second major type of function of UASs is to help people to find what they need in a form that they can deal with.

Beginning with ambitious dialog strategy:

S: Hi, this is AT&T Amtrak Schedule System. This is TOOT. How may I help you?

U: I need to find a train going to New York City.
(Interpretation by speech recognizer: "DC I don't care on Thursday".)

S: Which city do you want to leave from?

U: Baltimore.

S: Do you want me to find the trains from Baltimore to DC on Thursday at anytime now?

U: No

S: ...

After two switches to more conservative dialogue strategies:

S: Which city do you want to go to?

U: New York City.

S: Do you want to go to New York City?

U: Yes.

S: Which city do you want to leave from?

U: ...

Figure 15.8. *Excerpts from a dialog with* TOOT.
(Dialog excerpted and reformatted from Figure 2 of "Predicting and adapting to poor speech recognition in a spoken dialogue system," by D. J. Litman and S. Pan, 2000, in Proceedings of the Seventeenth National Conference on Artificial Intelligence, *Austin, TX, pp. 722–728. Copyright 2000 by the American Association for Artificial Intelligence.)*

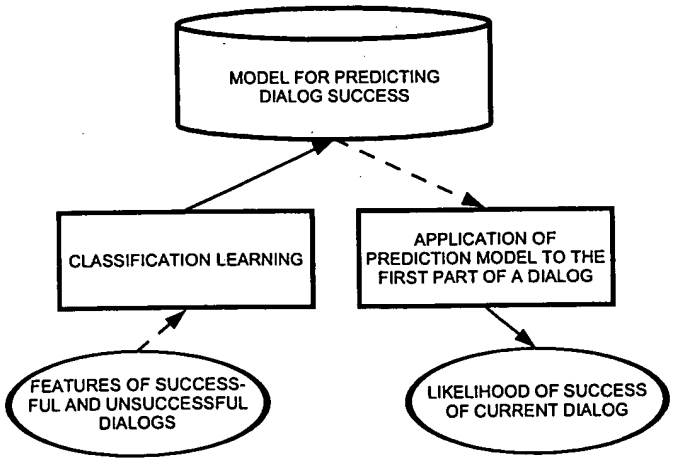

Figure 15.9. *Overview of adaptation in* TOOT.

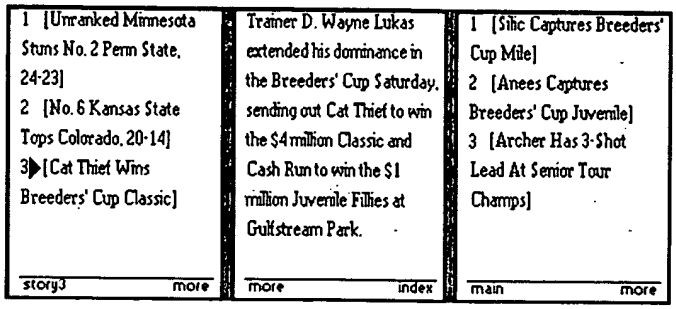

Figure 15.10. *Sequence of three screens presented by the* ADAPTIVE NEWS SERVER.
(Adapted from a slide supplied by Michael J. Pazzani. Copyright 2000 by Michael J. Pazzani.)

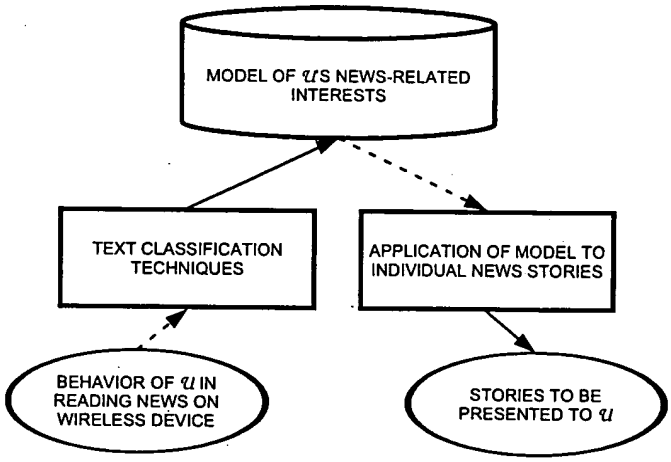

Figure 15.11. *Overview of adaptation in* ADAPTIVE NEWS SERVER.

15.3.1 Helping Users to Find Information

We will first look at the broad class of systems that help $\mathcal{U}$ to find relevant electronic documents, which may range from brief news stories to complex multimedia objects. A relatively novel example of such a system is the ADAPTIVE NEWS SERVER (Figure 15.10): This system delivers news stories to small, portable computing devices such as mobile phones and personal digital assistants. Screen A shows an overview of available news stories; the first two are about American football games, while the third is about a horse race. After $\mathcal{U}$ has selected and read the horse racing story (Screen B), instead of taking $\mathcal{U}$ back to Screen A, $\mathcal{S}$ generates an adapted screen (C): The first two stories now concern horse racing, a sport that $\mathcal{S}$ has inferred $\mathcal{U}$ to be interested in.

The systems in this category[2] typically draw from the vast repertoire of techniques for analyzing textual information (and to a lesser extent, information presented in other media) that have

[2]Surveys of parts of this large area are provided by, among others, Brusilovsky (1996, 2001), Hanani, Shapira, and Shoval (2001), and Mladenic (1999).

been developed in the field of information retrieval. The forms of adaptive support are in part different in three different situations:

Support for Browsing $\mathcal{U}$ may actively search for desired information by examining information items and pursuing cross-references among them, as in hypermedia systems such as the world-wide web or in the ADAPTIVE NEWS SERVER. A UAS can help focus $\mathcal{U}$'s browsing activity by recommending or selecting promising items or directions of search, on the basis of what $\mathcal{S}$ has been able to infer about $\mathcal{U}$'s information needs. Instead of selecting some documents at the expense of others, as in the ADAPTIVE NEWS SERVER, systems with more communication bandwidth typically highlight recommended hyperlinks and/or provide separate lists of recommendations. (Both of these methods are used, for example, in the WEBWATCHER system of Joachims, Freitag, & Mitchell, 1997.)

Support for Query-Based Search or Filtering Many systems provide some sort of query mechanism, such as a search engine or a keyword-based filtering mechanism. But explicit queries often only roughly reflect $\mathcal{U}$'s actual information need. A user model constructed on the basis of other sources of information can help to improve the selection of the documents presented to $\mathcal{U}$ and/or the appropriateness of the way in which they are presented (e.g., their sorting in terms of relevance). In one straightforward approach, which is used in the ADAPTIVE NEWS SERVER, $\mathcal{S}$ first processes a query in a nonadaptive way and then consults its model of $\mathcal{U}$'s interests when deciding about the further filtering and/or presentation of the results.

Spontaneous Provision of Information A number of systems present information that may be useful to $\mathcal{U}$ even while $\mathcal{U}$ is simply working on some task, making no effort to find information. For example, WATSON (Budzik, Hammond, & Birnbaum, 2001) monitors the text that a user is typing into a word processor and presents links to relevant documents in a web browser. (Other systems in this category are described by Maglio, Barrett, Campbell, & Selker, 2000, and by Rhodes, 2000.) The usability challenge of maintaining unobtrusiveness (see 15.4.3 below) is especially important here, since $\mathcal{U}$ is not actively searching for information (see Maglio & Campbell, 2000, for an experimental study of how best to achieve this goal).

15.3.2 Tailoring Information Presentation

Even in cases where it is clear which document a system should present to $\mathcal{U}$, the best specific way of presenting it may vary from one user to the next. Figure 15.12 shows part of a screen from the tourist information system AVANTI that describes a particular hotel in Siena, Italy (Fink, Kobsa, & Nill, 1998). $\mathcal{S}$'s current model of $\mathcal{U}$ states that $\mathcal{U}$ is interested in information for wheelchair-bound or dystrophic visitors; accordingly, information about the accessibility of several parts of the hotel is included. For other users, this information would be omitted on the grounds that it would only clutter the display. This type of variation in presentation can be realized to some extent through explicit adaptation by $\mathcal{U}$. Indeed, in the example $\mathcal{U}$ is given the opportunity to "Personalize the table below" by specifying which attributes are to be shown. But the limitations of pure adaptability that were discussed in connection with interface adaptation (15.2.2) require this approach to be combined with some spontaneous system adaptivity (cf. also the chapter by Stephanidis in this handbook). In the example in Figure 15.12, $\mathcal{S}$ has actually inferred from $\mathcal{U}$'s previous actions that $\mathcal{U}$ may no longer be interested in information for users with mobility limitations; $\mathcal{S}$ therefore gives $\mathcal{U}$ the option of allowing $\mathcal{S}$ to perform further adaptations to eliminate such information.

Personalize the table below.

Services offered	
Facilities:	*Credit cards accepted: Yes* *Groups allowed: Yes* *Animals allowed: Yes*
Laundry service:	*Yes, accessible*
Car service to station:	*No*
Car park:	*None*
Cash-register service:	*Yes, accessible*
Lift:	*Connects all floors*
Shared bathrooms:	*Yes*
Access to public areas:	*Presents no obstacles;* *3 stairs into the bar*

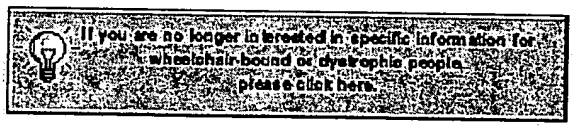

Figure 15.12. *Part of a screen from the* AVANTI *tourist information system.*
(Part of Figure 4 of "Adaptable and adaptive information provision for all users, including disabled and elderly people," by J. Fink, A. Kobsa, and A. Nill, 1998, New Review of Hypermedia and Multimedia, 4, *pp. 163–188. Copyright 2000 by Taylor Graham Publishers.)*

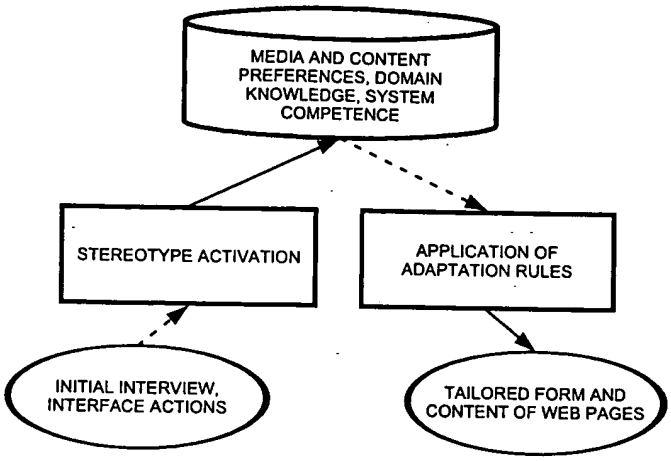

Figure 15.13. *Overview of adaptation in* AVANTI.

Evaluations of AVANTI confirmed that (a) mobility-impaired users appreciated the additional accessibility information that $\mathcal{S}$ made available and (b) user-controlled adaptation was taken advantage of mainly by experienced users of the system.

Another class of systems in which the tailoring of information to individual users can be especially beneficial comprises systems that present medical information to patients (see, e.g., Hirst, DiMarco, Hovy, & Parsons, 1997; Jones et al., 1999).

Properties of users that may be taken into account in the tailoring of documents include: $\mathcal{U}$'s degree of interest in particular topics; $\mathcal{U}$'s knowledge about particular concepts or topics; $\mathcal{U}$'s preference or need for particular forms of information presentation (for example, AVANTI generates textual descriptions in lieu of maps for visually impaired users); and the display capabilities of $\mathcal{U}$'s computing device (e.g., web browser vs. cell phone).

Even in cases where it is straightforward to determine the relevant properties of $\mathcal{U}$, the automatic creation of adapted presentations can require sophisticated techniques of natural language generation (see, e.g., Hirst et al., 1997) and/or multimedia presentation generation (see, e.g., André & Rist, 1995). Various less complex ways of adapting hypermedia documents to individual users have also been developed (see Brusilovsky, 1996, Section 6).

15.3.3 Recommending Products

One of the most practically important categories of UAS today comprises the product recommenders that are found in many commercial web sites—and also, increasingly, in mobile information servers.[3] A screen from typical film recommender system, the MOVIECENTRAL web site, is shown in Figure 15.14. The user has already rated 10 films, as is required by $\mathcal{S}$ before any recommendation can be made. On the basis of these ratings, $\mathcal{S}$ has identified a set of *neighbors* for $\mathcal{U}$: other users with tastes similar to $\mathcal{U}$'s. For the movie *2001, A Space Odyssey*, $\mathcal{S}$ has generated a "predicted rating" for $\mathcal{U}$ by examining the ratings of the subset of $\mathcal{U}$'s neighbors who have rated this movie. $\mathcal{U}$, who has seen this movie, has provided feedback by giving her actual rating of it. Another common form of adaptation, also shown in the figure, is the presentation of

[3] For a more general treatment of the HCI aspects of e-commerce, see the chapter by Vergo et al. in this handbook.

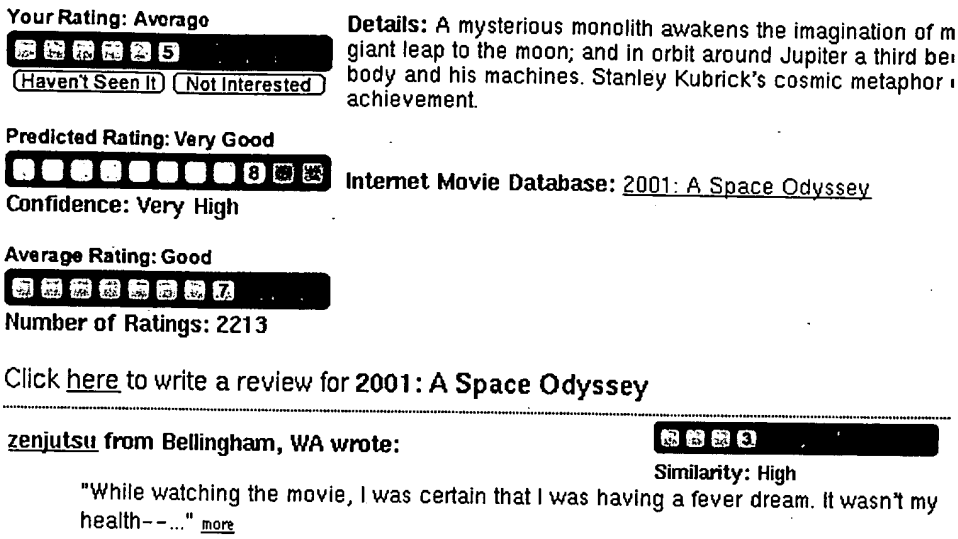

Figure 15.14. *Part of a screen from the* MOVIECENTRAL *film recommendation web site describing the movie "2001, A Space Odyssey".*
(Screen shot made from http://www.qrate.com/ in January 2001 and edited for compactness. This web site is no longer in operation.)

reviews that have been supplied by similar users. The general approach to adaptation summarized in Figure 15.14 is called *collaborative filtering* (see 15.6.2 for references and further discussion). Many studies have shown that collaborative filtering techniques produce recommendations whose accuracy for an individual user is usefully high.

Some product recommenders require $\mathcal{U}$ to specify her evaluation criteria explicitly instead of simply rating individual items. For example, if $\mathcal{U}$ is looking for a suitable dog with the help of PERSONALOGIC, the system will ask how important it is to $\mathcal{U}$ that the dog should be easy to train (cf. Figure 15.22 in 15.5.1 below). This method offers a natural alternative to collaborative filtering where relatively complex and important decisions are involved for which it is worthwhile for $\mathcal{U}$ to think carefully about the attributes of the products in question.

A third approach is exemplified by the FINDME family of recommenders (Burke, Hammond, & Young, 1997; Burke, 2000). The distinguishing feature is an iterative cycle in which $\mathcal{S}$ proposes a product (e.g., a restaurant in a given city), $\mathcal{U}$ criticizes the proposal (e.g., asking for a "more casual" restaurant), and $\mathcal{S}$ proceeds to propose a similar product that takes the critique into account. An advantage of this approach is that $\mathcal{U}$ receives the first recommendations very quickly. These recommendations, even if not especially suitable, can help $\mathcal{U}$ to clarify her own product evaluation criteria.[4]

Various combinations of these approaches, as well as specific other ideas, have been proposed (see, e.g., Burke, 2000; Schafer, Konstan, & Riedl, 1999).

Product recommenders address several problems that computer users typically experience when they search for products:

[4] At the time of this writing, a restaurant recommender from the FINDME family was available on the world-wide web at http://infolab.ils.nwu.edu/entree/pub/.

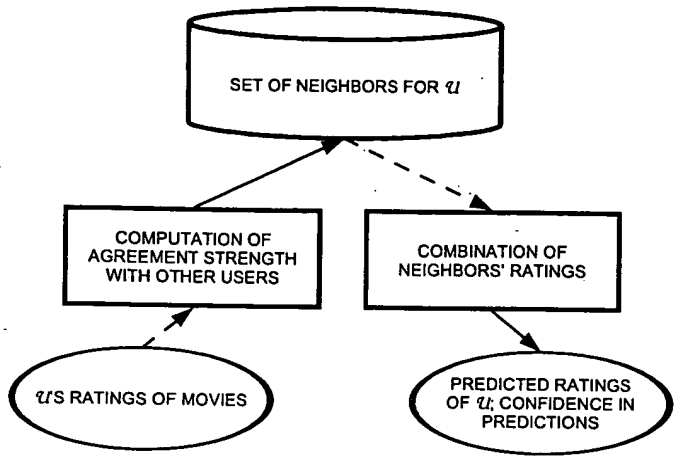

Figure 15.15. *Overview of adaptation in* MOVIECENTRAL.

1. $\mathcal{U}$ may not know what aspects of the products to attend to or what criteria should determine her decision. Some recommenders either (a) make it less necessary for $\mathcal{U}$ to be explicitly aware of her evaluation criteria (as when collaborative filtering is used) or (b) help $\mathcal{U}$ to learn about her own criteria during the course of the interaction with $\mathcal{S}$).
2. If $\mathcal{U}$ is unfamiliar with the concepts used to characterize the products, she may be unable to make effective use of any search or selection mechanisms that may be provided. Product recommenders generally reduce this communication gap by allowing $\mathcal{U}$ to specify her criteria (if this is necessary at all) in terms that are more natural to her.
3. $\mathcal{U}$ may have to read numerous product descriptions in various parts of the site, integrating the information found in order to arrive at a decision. Once a product recommender has acquired an adequate user model, $\mathcal{S}$ can take over a large part of this work, often examining the internal descriptions of a much larger number of products than $\mathcal{U}$ could deal with herself.

From the point of view of the vendors of the products concerned, the most obvious potential benefit is that users will find one or more products that they consider worth buying, instead of joining the notoriously large percentage of browsers who never become buyers (cf. Schafer et al., 1999). A related benefit is the prospect of cross-selling: $\mathcal{S}$'s model of $\mathcal{U}$ can be employed for the recommendation of further products that $\mathcal{U}$ might not have considered herself. Finally, some vendors aim to build up customer loyalty with recommenders that acquire long-term models of individual customers: If $\mathcal{U}$ believes that $\mathcal{S}$ has acquired an adequate model of her, $\mathcal{U}$ will tend to prefer to use $\mathcal{S}$ again rather than starting from scratch with some other system.

15.3.4 Supporting Collaboration

The increasing tendency for computer users to be linked via networks has made it increasingly feasible for users to collaborate, even in a spontaneous way and without prior acquaintance. A system that has models of a large number of users can facilitate such collaboration by taking into account the ways in which users match or complement each other.

A well-known example of a system of this sort is PHELPS (see, e.g., Collins et al., 1997; Greer et al., 1998). PHELPS is part of the Offender Management System used in the Correctional

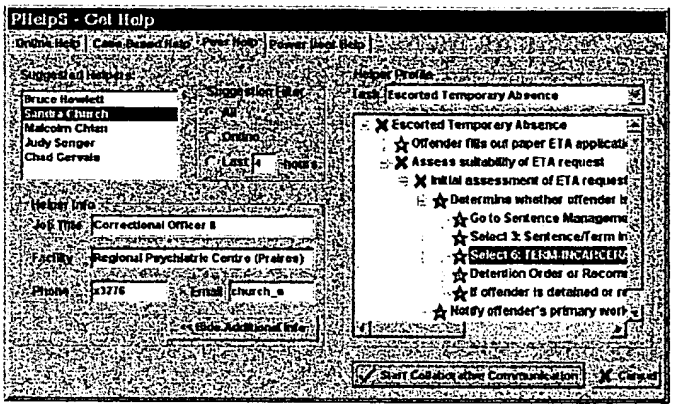

Figure 15.16. *Screen shot from the* PHELPS *system and overview of adaptation.*

(Left: U is having difficulty arranging an escorted temporary absence for a prisoner. S offers information on a number of possible helpers at various places in Canada. The window in the right-hand side of the screen shows a profile of a potential helper's knowledge of the task in question. Figure 2 of "Supporting peer help and collaboration in distributed workplace environments," by J. E. Greer, G. I. McCalla, J. A. Collins, V. S. Kumar, P. Meagher, and J. Vassileva, 1998, International Journal of AI and Education, 9, *pp. 159–177. Copyright 1998 by The International Artificial Intelligence in Education Society.)*

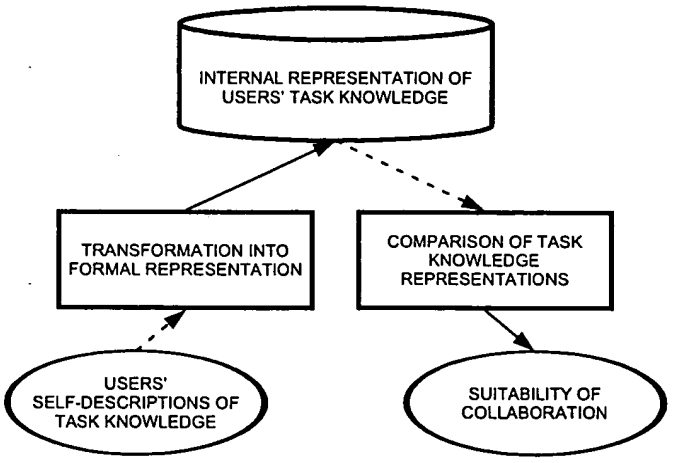

Figure 15.17. *Overview of adaptation in* PHELPS.

Services of Canada. It is designed to support relatively inexperienced workers (e.g., trainees) who are not sure how to handle a particular task that they are working on (see, e.g., the screen in Figure 15.16). The system suggests helpers who (a) have the relevant specific knowledge, (b) are available to provide help in the time frame required, (c) have not been overburdened with other help requests in the recent past; and (d) have other relevant positive characteristics (e.g., speaking the same language as $\mathcal{U}$). PHELPS was found to work as expected and to be accepted by representative potential users in a small-scale study with 4 trainees (Greer et al., 1998).

User modeling has been applied in connection with several (partially overlapping) types of collaboration:

- In computer-supported learning environments, in which the idea of *collaborative learning* has gained popularity in recent years (see, e.g., Paiva, 1997).
- As a way of providing "intelligent help" for complex tasks (see, e.g., Vivacqua & Lieberman, 2000). Putting a human expert into the loop is a way of avoiding some of the difficulties associated with in fully automatic adaptive help systems (15.2.3).
- In environments for computer-supported cooperative work within organizations (see, e.g., McDonald & Ackerman, 2000).

15.3.5 Supporting Learning

Research on *student modeling*—or *learner modeling*, as it has been called more often in recent years—aims to add user-adaptivity to computer-based tutoring systems and learning environments (cf. Corbett & Koedinger, 1997, and the chapter by Emurian & Durham in this handbook).[5]

Increasingly, learning environments are being made available on the world-wide web. An example is ELM-ART (Weber & Specht, 1997), which teaches users the programming language LISP. Figure 15.18 illustrates just one of the system's adaptive functions: the way in which it guides learners to parts of a course that it would be appropriate for them to study at a given time, given the skills that they have been observed to possess so far. $\mathcal{S}$ signals the suitability of learning units to $\mathcal{U}$ in two different ways, without restricting $\mathcal{U}$'s freedom to explore the learning environment on her own: (a) Link annotations (realized here as color-coded folders) indicate the extent to which a visit to a given unit is recommended. (b) The button at the bottom of the screen recommends a single unit that it would be especially appropriate for $\mathcal{U}$ to visit next.

Interaction in *intelligent tutoring systems* and *intelligent learning environments* can take many forms, ranging from tightly system-controlled tutoring to largely free exploration by the learner. In addition to navigation support (illustrated here by ELM-ART), aspects of the system that can be adapted to the individual user include: (a) the selection and the form of the instructional information presented; (b) the content of problems and tests; and (c) the content and timing of hints and feedback.

Learner modeling systems may adapt their behavior to any of a broad variety of aspects of the user, such as: (a) $\mathcal{U}$'s knowledge of the domain of instruction, including knowledge acquired prior to and during the use of $\mathcal{S}$; (b) $\mathcal{U}$'s learning style, motivation, and general way of looking at the domain in question; and (c) the details of $\mathcal{U}$'s current processing of a problem.

The underlying assumption is that the adaptation of $\mathcal{S}$'s behavior to some of these properties of the learner can lead to more effective and/or more enjoyable learning. One series of studies that directly demonstrates the added value of learner-adaptive tutoring is described by Corbett (2001). Many other evaluation studies assess the effectiveness of an entire system and therefore do

[5] Good sources of literature include the *International Journal of Artificial Intelligence in Education* and the proceedings of the biennial Conferences on Artificial Intelligence in Education (see, e.g., Lajoie & Vivet, 1999).

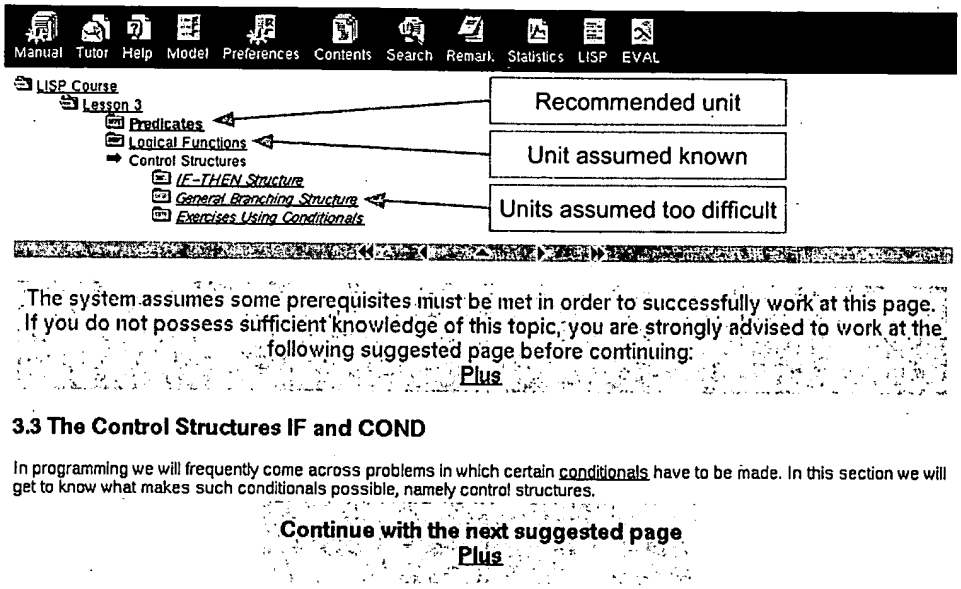

Figure 15.18. *Example screen from* ELM-ART *showing the system's assessment of the suitability of particular learning units for the current user.*

(Screen shot made from http://www.psychologie.uni-trier.de:8000/elmart in December 2000. In the actual system, the different colors of the folder icons are clearly distinguishable. Reproduced with permission of Gerhard Weber. [Permission has been requested].)

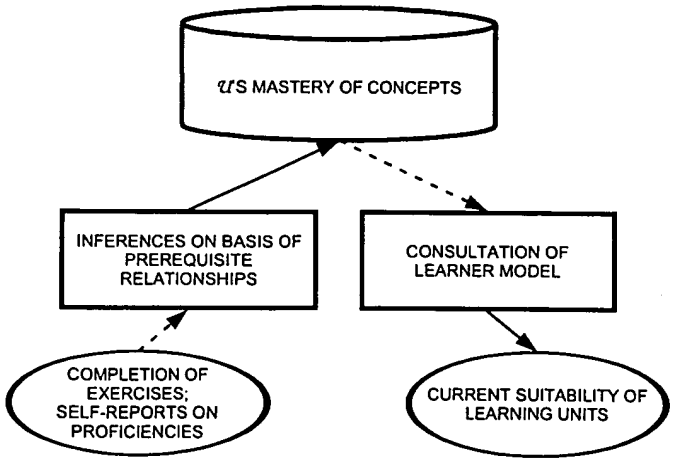

Figure 15.19. *Overview of adaptation in* ELM-ART.

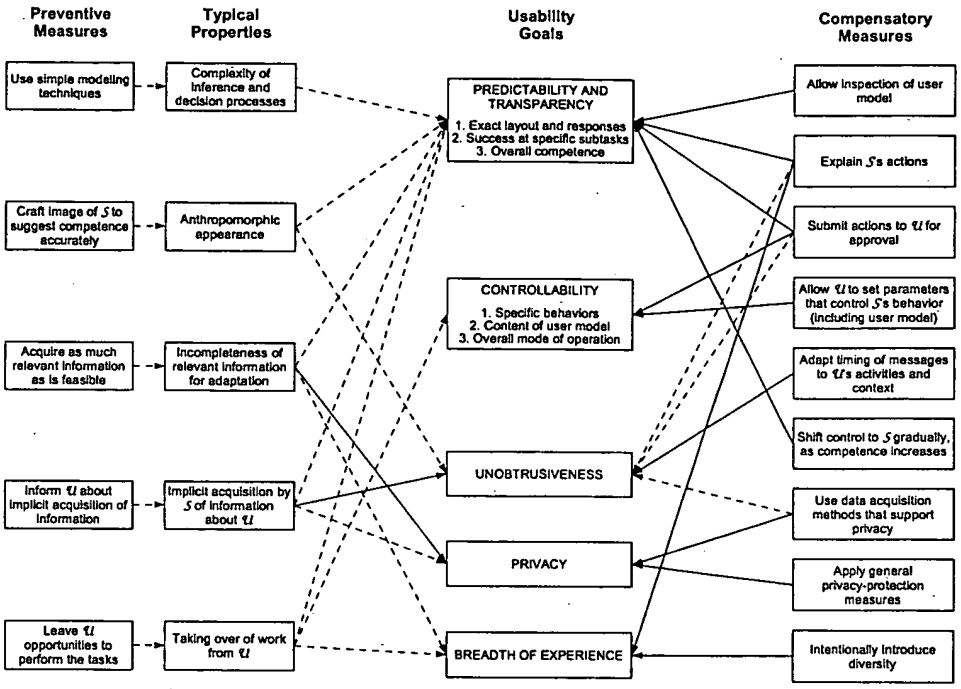

Figure 15.20. *Overview of usability challenges for user-adaptive systems.*
(Solid and dashed arrows denote positive and negative causal influences, respectively; further explanation is given in the text.)

not pinpoint the contribution of learner-adaptivity (see, e.g., Corbett, McLaughlin, & Scarpinatto, 2000, p. 91).

15.4 Usability Challenges

Some of the typical properties of user-adaptive systems can lead to usability problems that may outweigh the benefits of adaptation to the individual user. Discussions of these problems have been presented by a number of authors (see, e.g., Höök, 2000; Lanier, 1995; Norman, 1994; Schaumburg, 2001; Shneiderman, 1995; Wexelblat & Maes, 2001). Figure 15.20 gives a high-level summary of many of the relevant ideas.

The USABILITY GOALS shown in the third column correspond to several generally desirable properties of interactive systems. Those listed in the top three boxes (PREDICTABILITY AND TRANSPARENCY, INTERACTION QUALITY, and EFFICIENCY) correspond to general usability principles (see, e.g., the chapters by Stewart & Travis, by Cockton et al., and by van der Veer & Puerta Melguizo in this handbook. The remaining two goals, maintenance of PRIVACY and of BREADTH OF EXPERIENCE, are especially relevant to UASs.

The column TYPICAL PROPERTIES lists some frequently encountered (though not always necessary) properties of UASs, each of which has the potential of causing difficulties with respect to one or more of the usability goals.

Each of the remaining two columns shows a different strategy for ensuring that the usability goals are nonetheless realized: Each of the PREVENTIVE MEASURES aims to ensure that a typical property is not present in such a way that it would cause problems. Each of the COMPENSATORY MEASURES aims ensure in some other way that one or more goals are achieved despite the threats created by the typical properties.

A discussion of all of the relationships indicated in Figure 15.20 would exceed the scope of this chapter, but some remarks will help to clarify the main ideas.

15.4.1 Predictability and Transparency

The concept of *predictability* refers to the extent to which a user can predict the effects of her actions. *Transparency* is the extent to which she can understand system actions and/or has a clear picture of how the system works (cf. the chapter by van der Veer & Puerta Melguizo in this handbook). These properties are grouped together here because they are associated with largely the same set of other variables.

As the numbered items in the box PREDICTABILITY AND TRANSPARENCY indicate, users can try to predict and understand a system on several different levels of detail.

1. *Exact layout and responses.* Especially detailed predictability is important when interface elements are involved that are accessed frequently by skilled users—for example, icons in control panels or options in menus (cf. 15.2.2). If the layout and behavior of the system is highly predictable—in fact, essentially identical—over time, skilled users can engage in *automatic processing* (see, e.g., Hammond, 1987): They can use the parts of the interface quickly, accurately, and with little or no attention. In this situation, even minor deviations from complete predictability on a fine-grained level can have the serious consequence of making automatic processing impossible or error-prone.

2. *Success at specific subtasks.* Users may desire only more global predictability and transparency when $\mathcal{S}$ is performing some more or less complex task on $\mathcal{U}$'s behalf (e.g., searching for suitable products on the web): In the extreme case, $\mathcal{S}$ may want only to predict (or evaluate) the quality of the result of a complex system action.

3. *Overall competence.* The most global form of predictability and transparency concerns $\mathcal{U}$'s ability to assess $\mathcal{S}$'s overall level of competence: the degree to which $\mathcal{S}$ tends in general to perform its tasks successfully. With many types of system, high overall competence can be taken for granted; but as we have seen, the processes of acquiring and applying user models do not in general ensure a high degree of accuracy. If $\mathcal{U}$ seriously overestimates $\mathcal{S}$'s competence, she may rely on $\mathcal{S}$ excessively; if she underestimates $\mathcal{S}$, she will not derive the potential benefits that $\mathcal{S}$ can provide. A factor that is especially important with regard to this global level is the way in which the adaptive part of $\mathcal{S}$ is presented to $\mathcal{U}$. Some UASs (such as the OFFICE ASSISTANT, 15.2.3) have employed lifelike characters, for various reasons. As has often been pointed out, such anthropomorphic representations can invoke unrealistically high expectations concerning system competence—not only with regard to capabilities like natural language understanding but also with regard to $\mathcal{S}$'s ability to understand and adapt to $\mathcal{U}$.

In general, the levels and degrees of predictability and transparency that are necessary or desirable in a given case can depend on many factors, including the function that is being served by the adaptation and $\mathcal{U}$'s level of skill and experience. The same is true of the choice of the measures that are most appropriate for the achievement of predictability and transparency.

15.4.2 Controllability

Controllability refers to the extent to which $\mathcal{U}$ can bring about or prevent particular actions or states of $\mathcal{S}$ if she has the goal of doing so. Although controllability tends to be enhanced by transparency and predictability, these properties are not perfectly correlated. For example, when $\mathcal{U}$ clicks on a previously unused option in SMART MENUS (15.2.2), she can predict with certainty that it will be moved to the main part of its menu; but $\mathcal{U}$ has no control over whether this change will be made—except through the drastic step of deactivating the entire SMART MENUS mechanism.

A typical measure for ensuring some degree of control is to have $\mathcal{S}$ submit any action with significant consequences to $\mathcal{U}$ for approval. This measure may have negative impact on the usability goal of *unobtrusiveness* (see below); so it is an important interface design challenge to find ways of making recommendations in an unobtrusive fashion that still makes it easy for $\mathcal{U}$ to notice and follow up on them (cf. Figures 15.1, 15.6, and 15.12).

Like predictability and transparency, controllability can be achieved on various levels of granularity (see the items in the box labeled INTERACTION QUALITY in Figure 15.20). Especially since the enhancement of controllability can come at a price, it is important to consider what kinds of control will really be desired. For example, there may be little point in submitting individual actions to $\mathcal{U}$ for approval if $\mathcal{U}$ lacks the knowledge or interest required to make the decisions. Wexelblat and Maes (2001) recommend making available several alternative types of control for users to choose from.

15.4.3 Unobtrusiveness

We will use the term *obtrusiveness* to refer to the extent to which $\mathcal{S}$ places demands on the user's attention which reduce $\mathcal{U}$'s ability to concentrate on her primary tasks. This term—and the related words *distracting* and *irritating*—are often heard in connection with UASs. Figure 15.20 shows that (a) there are several different reasons why UASs can easily turn out to be obtrusive and (b) there are equally many corresponding strategies for minimizing obtrusiveness. Some of these measures can lead straightforwardly to significant improvements—for example, when it is recognized that distracting lifelike behaviors of an animated character are not really a necessary part of the system.

15.4.4 Privacy

User-adaptive systems typically (a) gather data about individual users and (b) use these data to make decisions that may have more or less serious consequences. Users may accordingly become concerned about the possibility that their data will be put to inappropriate use. Privacy concerns tend to be especially acute in e-commerce contexts (cf. 15.3.3; Ghosh & Swaminatha, 2001), and with some forms of support for collaboration (15.3.4), because in these cases (a) data about $\mathcal{U}$ are typically stored on computers other than the user's own, (b) the data often include personally identifying information; and (c) there may be strong incentives to use the data in ways that are not dictated by $\mathcal{U}$'s own interests. As will be discussed in Section 15.5, different means of acquiring information about users can have different consequences with regard to privacy. On the other hand, many of the measures that can be taken to protect privacy—for example, a policy of storing as little personally identifying data as possible—are not specific to UASs (see the chapters by Diller & Masten and by Friedman & Kahn in this handbook).

15.4.5 Breadth of Experience

When a UAS helps $\mathcal{U}$ with some form of information acquisition (Section 15.3), much of the work of examining the individual documents, products, and/or people involved is typically taken over by $\mathcal{S}$. A consequence can be that $\mathcal{U}$ ends up learning less about the domain in question than she would with a nonadaptive system (cf. Lanier, 1995). For example, if $\mathcal{S}$ recommends apartments in a given region to $\mathcal{U}$ (see, e.g., Burke et al., 1997; Shearin & Lieberman, 2001) and $\mathcal{U}$ simply accepts one of the recommendations, $\mathcal{U}$ may learn less about the real estate market in that region than she would if she took the trouble to search systematically through a real estate web site. One point of view here (see, e.g., Shneiderman & Maes, 1997, p. 53) is that it should be up to the user to decide whether she prefers to learn about a given domain or to save time by delegating work to a system. It may be worthwhile to give $\mathcal{U}$ a continuous spectrum of possibilities between complete control over a task and complete delegation of it. For example, many product recommendation systems allow users to alternate freely between pursuing $\mathcal{S}$'s recommendations and browsing through product descriptions in the normal way.

A second way in which adaptivity can narrow the user's experience is through excessive reliance on an incomplete user model. For example, suppose that an apartment seeker has so far shown interest only in apartments in the Bellvue area: If $\mathcal{S}$ accordingly supplies information only about other apartments in this area, $\mathcal{S}$ may never discover that $\mathcal{U}$ is willing to consider apartments in other areas. Some systems mitigate this problem by systematically proposing solutions that are *not* dictated by the current user model (see, e.g., Linden, Hanks, & Lesh, 1997; Shearin & Lieberman, 2001).

15.5 Obtaining Information About Users

Some of the usability challenges discussed in the previous section are closely connected with the ways in which information about individual users is acquired—a consideration which also largely determines the success of a system's adaptation. The next two subsections will look, respectively, at (a) information that $\mathcal{U}$ supplies to $\mathcal{S}$ explicitly for the purpose of allowing $\mathcal{S}$ to adapt; and (b) information that $\mathcal{S}$ obtains in some other way.

15.5.1 Explicit Self-Reports and -Assessments

Self-Reports About Objective Personal Characteristics

Information about objective properties of the user (such as age, profession, and place of residence) often has implications that are relevant for system adaptation—for example, concerning the topics that $\mathcal{U}$ is likely to be knowledgeable about or interested in. This type of information also has the advantage of changing relatively infrequently. Many UASs request information of this type from users, but the following caveats apply:

1. Specifying information such as profession and place of residence may require a fair amount of tedious menu selection and/or typing.

2. Since information of this sort can often be used to determine the user's identity, $\mathcal{U}$ may justifiably be concerned about privacy issues (cf. 15.4.4). Even in cases where such concerns are unfounded, they may discourage $\mathcal{U}$ from entering the requested information.

Some systems (e.g., the adaptive tour guide AMPRES, described by Rössel, 2000) address these problems by (a) restricting requests for personal data to the few pieces of information (if any) that $\mathcal{S}$ really requires; and (b) explaining the uses to which the data will be put. A novel approach was

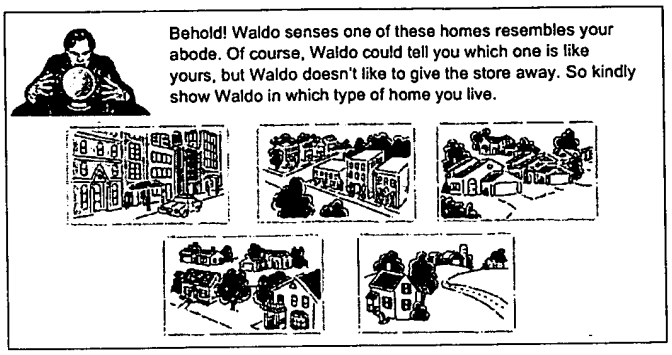

Figure 15.21. *Example of a screen with which the* LIFESTYLE FINDER *elicits demographic information.*

(Figure 3 of "Lifestyle Finder: Intelligent user profiling using large-scale demographic data," by B. Krulwich, 1997, AI Magazine, 18(2), pp. 37–45. Research conducted at the Center for Strategic Technology Research of Andersen Consulting (now Accenture Technology Labs). Copyright 1997 by the American Association for Artificial Intelligence.)

tried in the web-based LIFESTYLE FINDER prototype (Figure 15.21, Krulwich, 1997), which was characterized by a playful style and an absence of requests for personally identifying information. Of the users surveyed, 93% agreed that the LIFESTYLE FINDER's questions did not invade their privacy.

It is sometimes possible to avoid requests for explicit input about personal characteristics by accessing sources where similar information has already been stored (see 15.5.2).

Self-Assessments With Respect to General Dimensions

It is sometimes helpful for $\mathcal{S}$ to have an assessment of a property of $\mathcal{U}$ that can be expressed naturally as a position on a particular general dimension: the level of $\mathcal{U}$'s interest in a particular topic, the level of her knowledge about it, or the importance that $\mathcal{U}$ attaches to a particular evaluation criterion. Often an assessment is arrived at through inference on the basis of other evidence—for example, using the inference techniques discussed in 15.6.1 and 15.6.3. But it may be quicker, if feasible, to ask $\mathcal{U}$ for an explicit assessment. Figure 15.22 shows a typical rating scale. Some systems use checkboxes, which make it possible for $\mathcal{U}$ to give quick "yes/no" ratings for a larger number of dimensions (see, e.g., Pazzani & Billsus, 1999).

This assessment method raises basically the same problems that have long been familiar in fields, such as psychology and marketing research, in which questionnaires are regularly used. For example, users may not know the exact meaning of the various points on the scale, and they may be inclined to answer in a way that seems socially desirable (see, e.g., King & Bruner, 2000). Too often, the design of this type of scale for a UAS does not adequately take into account the methodological knowledge that has been built up with regard to such problems (see, e.g., the chapters by Blomberg et al., and by Karat in this handbook).

The effort involved in this type of self-assessment is more cognitive than physical, but it can still be enough to discourage users. So as with requests concerning personal characteristics, it is in general worthwhile to consider ways of minimizing such requests, making responses optional, and ensuring that the purpose is clear.

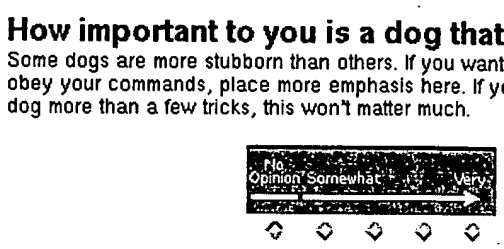

Figure 15.22. *Rating scale from the* PERSONALOGIC *decision guide for dog seekers.*
(Part of a screen shot made from http://www.purina.personalogic.com in July 2001. Before mid-2001 many similar recommenders had been available via http://www.personalogic.com.)

Self-Reports on Specific Evaluations

Instead of asking $\mathcal{U}$ to assess her position on a dimension explicitly, some systems try to infer $\mathcal{U}$'s position on the basis of her explicitly evaluative responses to specific items. For this purpose, $\mathcal{S}$ may present icons (e.g., "thumbs-up" and "thumbs-down"), checkboxes, or rating scales. The items that $\mathcal{U}$ evaluates can be (a) items that $\mathcal{U}$ is currently experiencing directly (e.g., the current web page), (b) actions that $\mathcal{S}$ has just performed (see, e.g., Billsus & Pazzani, 2000; Wolfman et al., 2001), (c) items that $\mathcal{U}$ must judge on the basis of a description (e.g., the abstract of a talk; a table listing the attributes of a physical product), or (d) the mere name of an item (e.g., a movie) that $\mathcal{U}$ may have had some experience with in the past (see, e.g., Figure 15.14). The physical effort required is usually that of a simple action such as a mouse click. The cognitive effort depends in part on how directly available the item is: In the third and fourth cases just listed, $\mathcal{U}$ may need to perform memory retrieval and/or inference in order to arrive at an evaluation.

Even when the effort is minimal, users often do not like to bother with explicit evaluations that do not constitute a necessary part of the task they are performing. One way of dealing with this problem is to develop inference methods that can make use of even a small number of explicit assessments of individual items (see Section 15.6). Another approach is to have $\mathcal{S}$ interpret naturally occurring actions of $\mathcal{U}$ (e.g., selection vs. skipping of news stories whose headlines are displayed on a screen) as implicit evaluations (cf. 15.5.2). These two approaches were evaluated by Billsus and Pazzani (2000) in two variants of the ADAPTIVE NEWS SERVER (15.3.1).

Responses to Test Items

In systems that support learning (15.3.5), it is often natural to administer tests of knowledge or skill. In addition to serving their normal educational functions, these tests can yield valuable information for $\mathcal{S}$'s adaptation to $\mathcal{U}$. An advantage of tests is that they can be constructed, administered, and interpreted with the help of a large body of theory, methodology, and practical experience (see, e.g., Wainer, 2000).

Outside of a learning context, users will in general be reluctant to invest time in tests of knowledge or skill, unless these can be presented in an enjoyable form (see, e.g., the color discrimination test used by Gutkauf, Thies, & Domik, 1997, to identify perceptual limitations relevant to the automatic generation of graphs). But it is sometimes possible to design a part of a system so that it functions as a concealed test, in addition to serving its primary function (e.g., of providing information). For example, the AMPRES hypertext system (Rössel, 2000) includes an introductory sequence in which $\mathcal{U}$ can request explanations of various concepts that are used within the system.

This sequence is designed so that the pattern of $\mathcal{U}$'s requests can be interpreted as if they were answers to items on a knowledge test.

15.5.2 Nonexplicit Input

Naturally Occurring Actions

The broadest and most important category of information about users includes all of the actions that $\mathcal{U}$ performs with $\mathcal{S}$ that do not have the express purpose of revealing information about $\mathcal{U}$ to $\mathcal{S}$. These actions may range from major actions like purchasing an expensive product to minor ones like scrolling down a web page. The more significant actions tend to be specific to the particular type of system that is involved (e.g., e-commerce sites vs. learning environments). Within some domains, there has been considerable research on ways of interpreting particular types of naturally occurring user actions. For example, researchers interested in adaptive hypertext navigation support have developed a variety of ways of analyzing $\mathcal{U}$'s navigation actions to infer $\mathcal{U}$'s interests and/or to propose navigation shortcuts (see, e.g., Goecks & Shavlik, 2000).

In their purest form, naturally occurring actions require no additional investment by the user, because they are actions that $\mathcal{U}$ would perform anyway. The main limitation is that they are hard to interpret; for example, the fact that a given web page has been displayed in $\mathcal{U}$'s browser for 4 minutes does not reveal with certainty which (if any) of the text displayed on that page $\mathcal{U}$ has actually read. Some designers have tried to deal with this tradeoff by designing the user interface in such a way that the naturally occurring actions are especially easy to interpret. For example, a web-based system might display just one news story on each page, even if displaying several stories on each page would normally be more desirable.

The interpretation of naturally occurring actions by $\mathcal{S}$ can raise privacy and transparency issues (cf. Figure 15.20) that do not arise in the same way with explicit self-reports and -assessments (15.5.1): Whereas the latter way of obtaining information about the user can be compared with interviewing, the former way is more like eavesdropping—unless $\mathcal{U}$ is informed about the nature of the data that are being collected and the ways in which they will be used (cf. Martin, Smith, Brittain, Fetch, & Wu, 2001).

Previously Stored Information

Sometimes a system can access relevant information about $\mathcal{U}$ which has been acquired and stored independently of $\mathcal{S}$'s interaction with $\mathcal{U}$:

1. If $\mathcal{U}$ has some relationship (e.g., patient, customer) with the organization that operates $\mathcal{S}$, this organization may have information about $\mathcal{U}$ that it has stored for reasons unrelated to any adaptation (e.g., $\mathcal{U}$'s medical record or address).
2. Relevant information about $\mathcal{U}$ may be stored in publicly available sources such as electronic directories or web homepages. For example, Pazzani (1999) explores the idea of using a user's web homepage as a source of information for a restaurant recommending system.
3. If there is some other system that has already built up a model of $\mathcal{U}$, $\mathcal{S}$ may be able to access the results of that modeling effort and try to apply them to its own modeling task. There is a line of research that deals with *user modeling servers* (see, e.g., Kobsa, 2001): systems that store information about users centrally and supply such information to a number of different applications. Some of the major commercial personalization software is based on this conception (see Fink & Kobsa, 2000, for an overview).

Relative to all of the other types of information about users, previously stored information has the advantage that it can in principle be applied right from the start of the first interaction of a

given user with a given system. To be sure, the interpretability and usefulness of the information in the context of the current application may be limited. Moreover, questions concerning privacy and transparency may be even more important than with the interpretation of naturally occurring actions.

Low-Level Indices of Psychological States

The next two categories of information about $\mathcal{U}$ have become practically feasible only in recent years, with advances in the miniaturization of sensing devices (cf. the chapter by Hinckley in this handbook).

The first category of sensor-based information (discussed at length by Picard, 1997) comprises data that reflect aspects of a user's psychological state, such as: (a) anger and frustration, which have especially clear relevance in the context of automated spoken dialogs with customers; (b) attraction to particular items, especially relevant for recommender systems; and (c) stress and cognitive load, which can be important factors when $\mathcal{U}$ is performing a challenging task (or several tasks at once), such as driving.

Two categories of sensing devices have been employed: (a) devices attached to $\mathcal{U}$'s body (or to the computing device itself) that transmit physiological data, such as electromyogram signals, the galvanic skin response, blood volume pressure, and the pattern of respiration; (b) video cameras and microphones that transmit psychologically relevant information about $\mathcal{U}$, such as features of her facial expressions (Picard, 1997) or of her speech (e.g., pitch, intensity, and quality of articulation, or more linguistic features such as the length of utterances and the occurrence of pauses—see Müller, Großmann-Hutter, Jameson, Rummer, & Wittig, 2001).

With both categories of sensors, the extraction of meaningful features from the low-level data stream requires the application of pattern recognition techniques. These typically make use of the results of machine learning studies in which the relationships between low-level data and meaningful features have been learned.

While it is sometimes possible to recognize a psychological state (such as anger) on the basis of sensor data alone, often this type of information needs to be combined with other types before reliable recognition is possible.

One advantage of sensors is that they supply a continuous stream of data, the cost to $\mathcal{U}$ being limited to the physical and social discomfort that may be associated with the carrying or wearing of the devices. These factors are significant now, but further advances in miniaturization—and perhaps changing attitudes as well—seem likely to reduce their importance.

Signals Concerning the Current Surroundings

As computing devices become more portable, it is becoming increasingly important for a UAS to have information about $\mathcal{U}$'s current surroundings (cf. the chapter by Stephanidis in this handbook). Here again, two broad categories of input devices can be distinguished:

1. Devices that receive explicit signals about $\mathcal{U}$'s surroundings from specialized transmitters. Some mobile systems that are used outdoors (see, e.g., Dey, Abowd, & Wood, 1998) employ GPS technology. More specialized transmitters and receivers are required, for example, if a portable museum guide system is to be able to determine which exhibit $\mathcal{U}$ is looking at.

2. More general sensing or input devices. For example, Schiele, Starner, Rhodes, Clarkson, and Pentland (2000) describe the use of a miniature video camera and microphone (each roughly the size of a coin) that enable a wearable computer to discriminate among different types of surroundings (e.g., a supermarket vs. a street). The use of general-purpose sensors eliminates the

dependence on specialized transmitters. On the other hand, the interpretation of the signals requires the use of sophisticated machine learning and pattern recognition techniques.

15.6 Learning, Inference, and Decision Making

A distinguishing feature of user-adaptive systems is the central role of techniques for user model acquisition and application (Figure 15.2). These techniques enable S to learn about individual users and make inferences and decisions about them. The present section describes the most important properties of several commonly used computational paradigms, which differ in terms of the contributions that they can make and the conditions under which they can be applied effectively.

15.6.1 Classification Learning

Many UASs employ learning methods from a broad category of machine learning techniques called *classification learning*. A great variety of methods have been developed within this paradigm, including: decision trees, probabilistic classifiers, neural networks, case-based reasoning, and specialized text-classification methods.[6]

From a broad perspective, the differences among these methods are less important than the basic nature of a classification learning problem, which will be illustrated with examples below: The learning procedure starts with a set of *training examples*, each of which is characterized in terms of its *features*. Each training example has been *classified*, that is, assigned to one of a set of two or more categories. On the basis of these examples, the procedure learns a *classifier*: a model that is capable of assigning a new item to one of the same set of categories. Usually the assignment for a new item cannot be made with certainty; accordingly, some methods yield a set of possible assignments for each item, each assignment being associated with some index of S's confidence.

Example Systems

These concepts are illustrated clearly by the SWIFTFILE system (Figure 15.1), which learns how to classify a user's email messages in (more or less) the same way as U would herself. At any given time, the training examples are the messages that U has filed so far, and the categories correspond to U's email folders. Once S has learned how to classify like U, S can *predict* how U will classify any given message—though of course not with perfect accuracy.

The particular type of model that SWIFTFILE learns takes advantage of the fact that each item to be classified (i.e., each email message) contains a large number of words that serve as features. SWIFTFILE employs well-established text classification methods from the information retrieval field to arrive at a representation of each email folder as a *weighted word-frequency vector*. To classify a new message, SWIFTFILE essentially compares the distribution of the words in its text with the current representations of U's folders (Segal & Kephart, 1999, 2000).

Here are some further examples of the use of classification learning for UASs:

1. Many systems that recommend or select documents (e.g., the ADAPTIVE NEWS SERVER, discussed in 15.3.1) employ some form of classification learning to learn to predict which documents a given U will like. The relevant features of the documents may include, in addition to the words in the text, attributes like length or date of appearance. Recommenders of this type are sometimes called *content-based* recommenders, because their predictions are based mainly on the

[6] Han and Kamber (2001), Langley (1996), and Mitchell (1997) offer broad overviews of statistical and machine learning techniques, some of which are related to those discussed in this section. The role of machine learning in UASs is reviewed by Webb, Pazzani, and Billsus (2001).

content of the documents in question, as opposed, say to the ratings of those items by other users (cf. 15.6.2 below).

2. CASPER (Bradley, Rafter, & Smyth, 2000) filters the search results returned by a job-finding web site by learning to predict which job offers the current $\mathcal{U}$ will like or dislike, on the basis of $\mathcal{U}$'s ratings of previous job offers. It makes use of the *nearest neighbor* classification method, and it employs a relatively sophisticated scheme for analyzing the features of job offers.

3. By observing how $\mathcal{U}$ enters appointments into a calendar system, the CALENDAR APPRENTICE (Mitchell et al., 1994) learns how to predict the properties of appointments that $\mathcal{U}$ makes, such as their location and duration. It employs the method of *decision tree induction*, which in effect yields rules for predicting the remaining features of an appointment on the basis of features that have already been specified.

Requirements

Two of the prerequisites for the application of classification learning methods may be hard to fulfill in some cases:

1. It may not be straightforward to characterize the items in question in terms of features. For example, when the items are images or music clips, it may be a challenging task to extract useful features (e.g., concerning content or style) from the digital representations of the items. When the items are nonelectronic objects to which the computer has no direct access (e.g., products offered for sale in an e-commerce site), information about their adaptation-relevant features must be (a) obtained from some existing information source or (b) entered specifically for the purpose of enabling adaptation.

2. It may not be possible for $\mathcal{S}$ to process an adequate number of training examples before it has to begin classifying new items. Depending on the nature of the system in question, users may or may not be willing to grant $\mathcal{S}$ ample training time before they expect useful adaptation to occur. There is currently a good deal of research into classification methods for UASs that can learn on the basis of a minimal number of examples. SWIFTFILE's text classification method rates well in this respect, and sometimes the nearest neighbor method (exemplified by the CASPER system) does so as well (see, e.g., Billsus & Pazzani, 2000). A different approach is to allow $\mathcal{U}$ to give $\mathcal{S}$ hints, in the form of explicit self-reports (15.5.1), that allow $\mathcal{S}$ to create an initial, partially accurate model that serves as a basis for further learning (see, e.g., Section 4 of Pazzani & Billsus, 1997).

15.6.2 Collaborative Filtering

The paradigm of *collaborative filtering* was illustrated in 15.3.3 with the movie recommender system MOVIECENTRAL. More generally, the approach is used for the prediction of the *responses* (e.g., ratings or purchases) of a user to *items* (e.g., documents or products) to which other users have previously responded. The distinguishing property of the paradigm is the fact that each item is characterized in terms of the previous responses of other users, not in terms of its intrinsic features.[7] Figure 15.23 summarizes a typical computational procedure.

Requirements

As with classification learning, certain prerequisites of the pure collaborative filtering paradigm are in some cases hard to fulfill:

[7] The less frequently used term *social recommendation* is on the whole more apt than *collaborative filtering*, since neither active collaboration nor filtering are essential aspects of the approach.

To identify neighbors for $\mathcal{U}$:

1. Store $\mathcal{U}$'s ratings of items.
2. For each user $\mathcal{U}^*$ in a sample of other users, compute an *agreement strength* with $\mathcal{U}$ on the basis of:
   - the difference between the ratings of $\mathcal{U}^*$ and $\mathcal{U}$;
   - the number of items rated by both $\mathcal{U}^*$ and $\mathcal{U}$.
3. Decide whether to add $\mathcal{U}^*$ as a neighbor on the basis of:
   - the agreement strength of $\mathcal{U}^*$ with $\mathcal{U}$;
   - the total number of items rated by $\mathcal{U}^*$;
   - the degree to which the total proportion of items rated by $\mathcal{U}$'s neighbors would be increased if $\mathcal{U}^*$ were added.

To make recommendations for $\mathcal{U}$:

1. For each item, use the ratings of $\mathcal{U}$'s neighbors to compute:
   - a predicted rating for $\mathcal{U}$;
   - a degree of confidence in this prediction;
   - the extent of disagreement among $\mathcal{U}$'s neighbors with regard to that item.
2. Base recommendations on these factors (among others).

Figure 15.23. *Summary of a typical algorithm for generating recommendations through collaborative filtering.*

1. For some of the items about which a prediction is to be made for $\mathcal{U}$, there may not be a sufficiently large number of responses available in the database that have been made to these items by users similar to $\mathcal{U}$ (or indeed by any users). For example, when new items continually enter the database and remain interesting for only a short time (e.g., news stories that are being fed in by a news service), they may have lost their importance by the time enough responses to them have been accumulated.

2. $\mathcal{U}$ may not be willing to give a sufficient number of responses to items before receiving useful recommendations. In particular, $\mathcal{U}$ may want to be able to specify a general preference explicitly, such as "I like science fiction movies". The only way for $\mathcal{U}$ to convey a preference like this in MOVIECENTRAL would be by (a) requesting the opportunity to rate a number of science fiction movies and (b) giving all of them high ratings.

Combinations With Other Paradigms

Some ingenious schemes have been devised for combining the basic strategy of collaborative filtering with other methods, so as to overcome some of the limitations of the pure form (see, e.g., Good et al., 1999, for an empirical comparison of a number of hybrid methods). Some researchers have explored ways of identifying suitable neighbors who have not necessarily responded to many of the same items as $\mathcal{U}$; instead, $\mathcal{S}$ can take into account their similarity to $\mathcal{U}$ in terms of personal characteristics (Pazzani, 1999) or interest profiles (Balabanovic & Shoham, 1997). Good et al. (1999) go a step further by introducing artificial "neighbors" who serve to fill in the gaps left by a user's human neighbors: Each such neighbor is an agent that implements a content-based prediction method, such as one that uses text classification methods (cf. 15.6.1).

15.6.3 Decision-Theoretic Methods

Basic Characteristics

A remarkable fact about the two paradigms discussed so far is that they are almost entirely *data-based*: They make use of virtually no general knowledge about users, their goals, or the items that they are dealing with. By contrast, the next paradigm represents a class of more *theory-based* methods: The system designers build into their models a good deal of knowledge about the variables that are relevant in a given interaction situation.

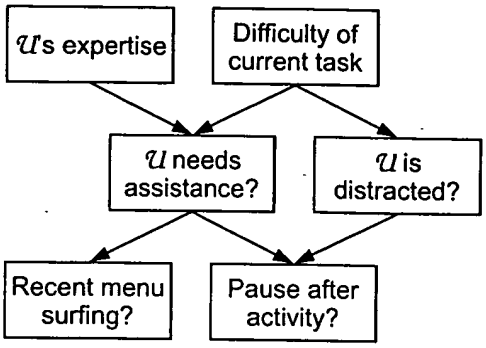

Figure 15.24. *Part of a Bayesian network used in the* LUMIÈRE *prototype for inferring the likelihood that $\mathcal{U}$ requires assistance.*

(Adapted from Figure 1 of "The Lumière project: Bayesian user modeling for inferring the goals and needs of software users," by E. Horvitz, J. Breese, D. Heckerman, D. Hovel, and K. Rommelse, 1998, in G. F. Cooper & S. Moral (Eds.), Uncertainty in Artificial Intelligence: Proceedings of the Fourteenth Conference, pp. 256–265, San Francisco: Morgan Kaufmann. Copyright 1998 by Morgan Kaufmann Publishers.)

For example, the partial *Bayesian network* in Figure 15.24 shows a few of the assumptions made by the designers of the LUMIÈRE adaptive help prototype (15.2.3).[8] . Each rectangle represents a variable about which $\mathcal{S}$ in general has only an uncertain belief. For example, the probability that the value of u NEEDS ASSISTANCE? is *true* at a given moment might be estimated as 89%, as in the situation illustrated in Figure 15.6. Arrows represent probabilistic relationships among the variables, which can usually be interpreted as causal influences. For example, this network states that $\mathcal{U}$'s need for assistance from $\mathcal{S}$ will depend largely on $\mathcal{U}$'s expertise in using $\mathcal{S}$ and on the difficulty of the task that $\mathcal{U}$ is currently dealing with. In turn, $\mathcal{U}$'s need for assistance will influence the occurrence of behaviors of $\mathcal{U}$ such as surfing through menus and/or pausing after performing some actions.

Sometimes $\mathcal{S}$ obtains information that leads to certainty about the value of a particular variable. This new certain belief is then propagated through the Bayesian network according to an applicable algorithm, and $\mathcal{S}$'s beliefs concerning the other variables in the network are adjusted accordingly. For example, when a "pause after activity" is observed, the probabilities associated with u NEEDS ASSISTANCE? and u IS DISTRACTED? will in general increase.[9]

Potential Advantages

Relative to the simpler data-based models discussed in the previous two sections, theory-based Bayesian networks offer several potential advantages, including the following:

1. $\mathcal{S}$ can make useful inferences about a user $\mathcal{U}$ without first acquiring any long-term model of $\mathcal{U}$ on the basis of data. For example, $\mathcal{S}$ might offer the advice shown in Figure 15.6 after observing just a few of $\mathcal{U}$'s actions within a spreadsheet application.
2. With the help of other closely related decision-theoretic techniques, the probabilistically

[8] A good introductory overview Bayesian networks and other decision-theoretic techniques is given by Jensen (2001); the classic work is by Pearl (1988). An early survey of applications within UASs is given by Jameson (1996), and recent developments are discussed by Zukerman and Albrecht (2001).

[9] More complex, application-specific Bayesian networks are used to estimate what particular help topics are most likely to be relevant to $\mathcal{U}$; see Figure 3 of Horvitz et al. (1998).

expressed beliefs generated by a Bayesian network can be used to make adaptation *decisions*. Such a decision-making process quantitatively takes into account the possible consequences of $\mathcal{S}$'s actions and the overall *utility* of these consequences. For example, $\mathcal{S}$ can quantitatively weigh the expected benefits of offering advice against the possible costs in terms of time and distraction. Note that the two paradigms discussed above—classification learning and collaborative filtering—yield predictions about a user (e.g., about the films that $\mathcal{U}$ is most likely to enjoy); they do not yield decisions as to what $\mathcal{S}$ should do (e.g., how many films should be recommended to $\mathcal{U}$, and in what form?). With these paradigms, therefore, the system designer must specify in advance how $\mathcal{S}$'s predictions are to be translated into $\mathcal{S}$'s actions. By contrast, decision-theoretic methods allow $\mathcal{S}$ to select actions more flexibly, taking into account $\mathcal{U}$'s perceived priorities and the details of the current situation (cf. Jameson et al., 2001). To be sure, this greater flexibility may sometimes diminish the predictability and transparency of $\mathcal{S}$'s actions (cf. 15.4.1).

Requirements

The major challenge in the development of decision-theoretic UASs concerns the construction of suitable general models, which are almost always much more complex than the partial model shown in Figure 15.24. If the construction is purely theory-based, one or more persons with relevant knowledge must specify the qualitative and quantitative relationships among the variables of the model, some of which are typically unobservable. It is in general difficult to make well-founded judgments about all aspects of such a model. In recent years, many techniques have been developed for learning decision-theoretic models at least partly on the basis of data (see, e.g., Heckerman, 1998). In connection with UASs, these methods have been applied successfully with relatively simple models (see, e.g., b=MuellerGJ+01)that are similar to the types of data-based model that were discussed in 15.6.1 and 15.6.2. The question of how to learn more complex, theoretically interpretable decision-theoretic models at least partly on the basis of data is a challenge that is being addressed in current research (see, e.g., Wittig & Jameson, 2000).

15.6.4 Other Approaches

Two other largely theory-based paradigms are worth mentioning briefly, although they are at present somewhat less widely used than the three paradigms discussed so far.

Techniques for Plan Recognition

In the field of artificial intelligence, many approaches have been developed to the problem of recognizing a person's plans on the basis of her observed actions. For UASs, these techniques offer the possibility that a system can interpret a user's actions as steps in the execution of a plan that is intended to achieve some goal; $\mathcal{S}$ may then be able to assist $\mathcal{U}$ in various ways, which depend in part on the function that the adaptation is intended to serve:

1. Some systems that aim to take over routine actions from $\mathcal{U}$ (15.2.1) do so by (a) recognizing a plan that $\mathcal{U}$ needs to execute repeatedly and (b) offering to execute the plan for $\mathcal{U}$ in future situations. This category includes some systems for *programming by example* (see, e.g., Lieberman, 2001).
2. Help systems (15.2.3) and tutoring systems (15.3.5) may point out problems with $\mathcal{U}$'s plan or remind $\mathcal{U}$ of steps that need to be taken.
3. Systems that conduct dialogs (15.2.4) can choose their own dialog contributions in accordance with $\mathcal{U}$'s perceived plan.

An overview of uses of plan recognition in interactive systems is given by Carberry (2001).

The Stereotype Approach

A stereotype–based system (see, e.g., Rich, 1989) distinguishes a set of categories, called *stereotypes*, that a given user may belong to. For example, in the AVANTI tourist information system (15.2.2), each stereotype corresponds to a group of visitors that have a certain set of capabilities and information needs. The system provides a set of rules for assigning each user to one or more stereotypes on the basis of $\mathcal{U}$'s observed behavior or other information about $\mathcal{U}$ (such as self-reports). Once $\mathcal{U}$ has been categorized in this way, $\mathcal{S}$ can ascribe to $\mathcal{U}$ properties and/or take actions that are associated with the stereotype(s) in question.

The stereotype approach was the first inference paradigm to be widely used for UASs (see, e.g., Rich, 1979). It is currently employed in some commercial personalization servers (cf. Fink & Kobsa, 2000), among other systems.

The inference processes in stereotype-based systems can be realized with a variety of computational techniques. The emphasis is less on sophisticated computation than on realistic specification of the content of the stereotypes and the rules for activating them.

15.7 Empirical Methods

Like any other type of interactive computing system or device, a UAS cannot be designed on the basis of first principles alone. No matter how sophisticated the techniques that are used to realize the system, what ultimately counts is how well the larger "system" that includes the user(s) works. So at various points in the design process, some sort of empirical work will need to be done to ensure that the design is in touch with reality.

The full repertoire of empirical methods in human-computer interaction (cf. Section VI of this handbook) is in principle applicable to user-adaptive systems. This section will focus on four categories of empirical study that are especially relevant and/or raise some important general issues when they are applied to UASs.[10] With each type of study, we will consider what it can tell us about each of the following two questions:

1. Accuracy of modeling: One important difference between UASs and other interactive systems is that a UAS typically derives testable hypotheses about each individual user $\mathcal{U}$. It is therefore often worthwhile to ask to what extent $\mathcal{S}$'s modeling of $\mathcal{U}$ is accurate. First, reasonable accuracy of $\mathcal{S}$'s modeling is in general a necessary (though not sufficient) condition for the success of $\mathcal{S}$'s adaptation. Second, it can be hard to know the implications of overall usability results if the accuracy of the modeling is not known—unless the results turn out to be conclusively positive.

2. Meeting usability challenges: The general usability challenges discussed in Section 15.4 constitute one reason why it is important not to restrict empirical studies to the question of modeling accuracy.

15.7.1 Wizard-of-Oz Studies

Systems that adapt to their users are in one methodological respect similar to systems that make use of speech (cf. the chapter by Lai & Yankelovich in this handbook): They attempt to realize a capability that is so far possessed to the highest degree by humans. Consequently, as with speech interfaces, valuable information can sometimes be obtained from a *Wizard-of-Oz study*: In a specially created setting, a human takes over a part of the processing of the to-be-developed

[10] More extended discussions of empirical methods for UASs are provided by Langley and Fehling (1998), Höök (2000), and Chin (2001).

Method

Subjects

- Were told an experimental help system would track their activity and make guesses about how to help them.
- Received the advice via a computer monitor.

Experts

- Worked in a separate room.
- Viewed subjects' activity via a monitor.
- Conveyed advice by typing.
- Were not informed about the assigned spreadsheet tasks.

Results

Difficulty of experts' task

- Experts showed some ability to identify $\mathcal{U}$'s goals and needs.
- They were often uncertain about:
  1. $\mathcal{U}$'s goals – sometimes recognized with an "Aha!" reaction after a period of confusion;
  2. the value of providing different kinds of assistance.

Consequences of poor advice

- Users typically examined advice carefully.
- Even when advice was off the mark, subjects would often become distracted by it and begin to experiment with the features described.
- This behavior gave experts false confirmation of successful goal recognition.
- Experts then gave further advice along the same lines.

How experts improved

- Experts became more skillful in offering advice in this situation.
- For example, they learned to give conditional advice: "If you are trying to do *X*, then ...."

Figure 15.25. *Summary of a Wizard-of-Oz study conducted in the* LUMIÈRE *research project.* *(Summarized on the basis of p. 258 of Horvitz et al., 1998.)*

system $\mathcal{S}$ for which humans are especially well suited (cf. the chapters by Lai & Yankelovich, by Beaudouin-Lafon & Mackay, and by Pew in this handbook).

The left-hand side of Figure 15.25 summarizes a typical study of this type that was conducted in an early phase of the development of the LUMIERE prototype (cf. 15.2.3). Whereas the users believed they were interacting with an adaptive help system, the help was actually provided by usability experts.

Assessing Accuracy A Wizard-of-Oz study can yield an *upper-bound* estimate of the highest level of modeling accuracy that might be attainable given the available information—as long as one can assume that the human "wizards" are more competent at the type of assessment in question than a fully automatic system is likely to be in the foreseeable future. For example, if the expert advisors in this particular study had shown no ability at all to recognize the users' goals, perhaps the entire project would have been reconsidered.

Assessing Usability The example study brought to light a subtle problem: the tendency of the advisors' recommendations to become self-fulfilling prophecies. This problem is related to the general usability challenge concerning *breadth of experience* (15.4.5). Here, however, the limited accuracy of the experts' "user models" did not lead to a narrowing of the users' experience; instead it caused users to be led into new areas that were presumably of no real interest to them. Note that the experts' behavior in this study also suggested a way of avoiding this problem. Of course any study in which the system is simulated by a human will reveal little about usability issues that involve details of the appearance and behavior of the user interface—such as the behavior of the animated characters that personify the OFFICE ASSISTANT.

15.7.2 Simulations Using Data From a Nonadaptive System

This second type of empirical study is uniquely applicable to user-adaptive systems. It focuses entirely on the issue of accurate modeling of $\mathcal{U}$. A typical basic procedure is as follows:

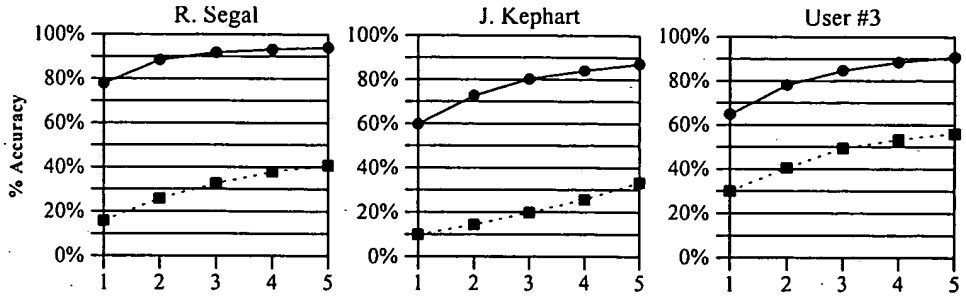

Figure 15.26. *Comparison of alternative variants of* SWIFTFILE *on the basis of data concerning the use of a nonadaptive system.*

(Upper curves: Accuracy of SWIFTFILE*'s predictions with different numbers* N *of suggestion buttons, shown on the x-axis. Lower curves: Accuracy of the naive strategy that simply predicts the* N *most frequently used folders. Part of Figure 3 of "MailCat: An intelligent assistant for organizing e-mail," by R. B. Segal and J. O. Kephart, 1999, in* Proceedings of the Third International Conference on Autonomous Agents, *pp. 276–282. Copyright 1999 by the Association for Computing Machinery, Inc.)*

Given a database of behavioral data on how a number of users have used a *nonadaptive* version of a system $\mathcal{S}$, simulate a use situation in which $\mathcal{S}$ receives parts of these data incrementally as information about $\mathcal{U}$, checking how fast and how well $\mathcal{S}$ can acquire and apply a model of $\mathcal{U}$.

A clear example of a study of this type was conducted with the SWIFTFILE system (Segal & Kephart, 2000). The training examples with respect to each $\mathcal{U}$ were $\mathcal{U}$'s previously filed messages. SWIFTFILE's learning method was applied to these messages just as if the system had done the learning while $\mathcal{U}$ was originally filing them. The overall result of the study was that $\mathcal{S}$ quickly and consistently attained high accuracy in predicting the folder that $\mathcal{U}$ would choose, including the correct folder about 90% of the time in its three guesses for each message. To be sure, this result applies only if the messages for which $\mathcal{U}$ decided to create a new folder are not counted, which is a reasonable policy.

Moreover, it is possible to evaluate many alternative variants of $\mathcal{S}$ on the basis of the same data, whereas each user could work with only one variant at any given time. Figure 15.26 shows the results of an analysis of this type. The results indicate how often SWIFTFILE would have suggested the correct folder, on the average, using each possible number of suggestion buttons between 1 and 5. On the basis of these results, Segal and Kephart (1999) argue that the optimal number of buttons is 3. The results also allow us to reject the hypothesis that $\mathcal{S}$ could do just as well by always suggesting the N most commonly used folders.

As these examples show, precise, thorough analyses can be performed without any investment of users' time—and without waiting for enough data from each user to accumulate, which could take months in the case of an email system.

Assessing Accuracy In some respects, the accuracy estimates that can be obtained with data from a nonadaptive system are actually more realistic than those that could be obtained in studies involving interaction with an adaptive system: When using the real SWIFTFILE system, users might sometimes be inclined to accept one of $\mathcal{S}$'s folder suggestions even when they would normally have chosen some other folder: $\mathcal{U}$ can save time and mental effort by relying uncritically on $\mathcal{S}$'s classifications. In this case the predictions of $\mathcal{S}$ would be self-fulfilling prophecies to some extent—like some of the advice of the experts in the Wizard-of-Oz study discussed in 15.7.1; and

the resulting accuracy estimates would be inflated to some unknown extent. Simulation studies can avoid the problem of self-fulfilling prophecies. Still, they should not be relied on exclusively, since the self-fulfilling prophecy phenomenon is one that can arise with real system use. The most complete picture is given by a combination of simulation studies and studies of real use—a procedure followed by Segal and Kephart, who also report on a study of actual system use (Segal & Kephart, 1999, Section 6).

Assessing Usability Simulations with data from a nonadaptive system cannot yield information about how users actually interact with a corresponding adaptive system. Therefore, they do not directly address the typical usability challenges discussed in Section 15.4. At best, results about $\mathcal{S}$'s modeling accuracy can form a basis for speculation about particular aspects of $\mathcal{S}$'s usability, such as $\mathcal{S}$'s predictability.

15.7.3 Controlled Studies

The third category of study is one which is familiar from many evaluation studies in the human-computer interaction field: Two or more variants of a given system are compared within a controlled setting in which users perform more or less predefined tasks. When a UAS is involved, typically an adaptive and a nonadaptive variant of the system are compared. With the current state of the art, the key question is often whether the adaptive version shows any advantages. In a few years, we may see more studies in which the main focus is on the comparative evaluation of different adaptive variants.

The evaluation of the dialog system TOOT (15.2.4; Litman & Pan, 2000) provides examples of the types of information that can be obtained from a controlled comparison. Six novice users conducted dialogs with an adaptive version of TOOT that was able to change its dialog strategy in the way illustrated in Figure 15.8, starting with the most ambitious strategy. Six other users conducted dialogs with a nonadaptive version that always used the most ambitious strategy.

The central result of the study concerns the likelihood that a given dialog will end successfully, with $\mathcal{U}$ hearing the schedule information for the desired train. The likelihoods were 65% and 23% for the adaptive and nonadaptive versions, respectively, the difference being statistically significant. Taken by itself, this result would simply confirm that a system that often shifts to a more long-winded, conservative dialog strategy will have a higher likelihood of successful dialog completion. But the dialogs conducted with the adaptive version were actually somewhat shorter than those with the nonadaptive version (though the difference in length was not statistically significant).

These results show that the adaptive version of TOOT was clearly preferable to the nonadaptive version used in this study. But as Litman and Pan acknowledge, it is still quite possible that some other nonadaptive version could outperform the adaptive version by consistently applying a single, more conservative dialog strategy. The general point is that it can be difficult to demonstrate that user-adaptivity yields the best results in a given situation. It is not enough to show that a particular nonadaptive version of the system performs less well than an adaptive version. The real question is whether *any* single nonadaptive system could feasibly be identified in advance that could do as well as an adaptive one.

This point is reminiscent of the lessons learned from attempts to evaluate the relative merits of competing user interface paradigms through empirical comparisons: No matter how flawless the methodology may be, it is hard to generalize the results with confidence beyond the particular system variants, tasks, and user groups that figured in the study.

Assessing Accuracy One typical obstacle to the assessment of modeling accuracy on the basis of data concerning system use was already discussed in the previous subsection: the problem of potentially self-fulfilling prophecies. The adaptation in TOOT illustrates a still more fundamental obstacle: TOOT switches to a new dialog strategy when it predicts that the current strategy will not lead to success; but once the switch has occurred, there is no way of determining what would actually have happened with the original strategy.

Assessing Usability Controlled studies offer an opportunity to compare users' subjective ratings of two or more system variants. In addition to overall satisfaction, these ratings may include assessments of the key usability variables predictability, transparency, controllability, and unobtrusiveness (cf. Section 15.4). Generalizing such ratings to situations of real use is problematic, however. Various factors that may strongly influence $\mathcal{U}$'s desire for adaptive features may be absent in the controlled setting, such as: time pressure, personal significance of the tasks performed with $\mathcal{S}$, and distractions due to competing tasks and events. These contextual factors can be taken into account in studies of actual use.

15.7.4 Studies of Actual System Use

The final major category of user studies is likewise familiar from its frequent use with nonadaptive systems: Some more or less complete version of $\mathcal{S}$ is employed by users in a more or less realistic setting; and various objective and/or subjective variables are assessed.

Observation and Interviewing The typical field study of a UAS involves a prototype that is used by several people in their normal everyday setting. Sawhney and Schmandt (2000, pp. 377–380) used this method to evaluate their NOMADIC RADIO, a wearable system for the context-sensitive transmission of audio messages. They observed and interviewed two experienced users of mobile phones and pagers who tried out a prototype of the NOMADIC RADIO during a three-day period. Although this type of study cannot rigorously test general hypotheses, the study did yield numerous suggestions that led to design improvements. Most of these concerned the way in which the system fit into the users' everyday patterns of work and social interaction. One result concerned $\mathcal{S}$'s assignment of priorities to messages, which it used to determine the nature and timing of notification: The users found that explicit, distinct auditory cues to a message's priority were less useful than simply having a message presented in a manner appropriate to its priority. (Note that what is involved here is a tradeoff between transparency and unobtrusiveness; cf. Figure 15.20). More generally, one user found the original scheme of auditory cues to be too complex to be used effectively during everyday activities and interactions. As was noted above, it would be more difficult to obtain useful results concerning questions like this from a controlled study.

Use of Questionnaires More extensive data on actual use—including use over a long period of time—can be obtained through questionnaires. For example, Schaumburg (2001) obtained data from 105 largely experienced users of the OFFICE ASSISTANT (cf. 15.2.3), each of whom spent 10–15 minutes filling out a questionnaire. Subjects reported both on their actual use of the system (e.g., whether they tended to use it when confronted with a problem) and on their subjective reactions to it (in particular, evaluations concerning key usability variables).

Assessing Accuracy Field studies can yield information about modeling accuracy if it is possible after the fact to check S's predictions against U's actual behavior—for example, when S makes recommendations which U can either accept or reject (see, e.g., the interesting analysis of this type conducted by Mitchell et al., 1994). But as was discussed above (15.7.2), it may be at least as effective to employ data concerning the use of a nonadaptive version of the system for this purpose.

Assessing Usability As the examples given have indicated, studies of actual use yield especially valuable information concerning the bottom-line question of the overall usefulness and usability of a UAS in real situations. If the study is designed accordingly, comparisons between competing system versions can be made (see, e.g., the large-scale field trial of a personalized medical information system described by Jones et al., 1999). As always with empirical studies, care must be taken in generalizing beyond the particular system versions, user groups and tasks employed in the study.

15.8 The Future of User-Adaptive Systems

This chapter has shown that adaptive interfaces, agents, and other user-adaptive systems do not represent a smooth and easy road to more successful human-computer interaction: They present a complex set of usability challenges (Section 15.4); they require carefully designed methods of acquiring information about users (Section 15.5), as well as relatively sophisticated computational techniques that are not needed in other types of interactive system (Section 15.6). And even when all of these requirements have been dealt with, it is often tricky to prove empirically that user-adaptivity has actually added any value (Section 15.7). It is no wonder that some experts believe that the interests of computer users are better served by continued progress within more familiar paradigms of user-centered system design (see, e.g., Shneiderman & Maes, 1997).

On the other hand, our understanding of the complex challenges raised by user-adaptive systems has been growing steadily, and they are now familiar and valued elements in a number of types of system, as the survey in Sections 15.2 and 15.3 has shown.

15.8.1 Growing Need for User-Adaptivity

Increases in the following variables suggest that the functions served by user-adaptivity will continue to grow in importance:

Diversity of Users and Contexts of Use As several chapters in this handbook make clear, computing devices are being used by an ever-increasing variety of users in an increasing variety of contexts. (See Section B, *Interaction Issues for Diverse Users*, as well as the chapter by Stephanidis & Savidis.) It is therefore becoming harder to design a system that will be suitable for all users and contexts without some sort of user-adaptivity or user-controlled adaptability; and as has been discussed at several points in this chapter (15.2.2, 15.3.2), adaptability has its limitations.

Number and Complexity of Interactive Systems The functions of user-adaptivity discussed in Section 15.2 partly involve helping users to deal effectively with interactive systems and tasks even when they are not able or willing to gain complete understanding and control in each individual case. This goal becomes increasingly important as the number—and in some cases the

complexity—of the systems that people must deal with continues to increase—because of factors ranging from the growth of the world-wide web to the proliferation of miniature interactive computing devices.

Scope of Information to Be Dealt With Even when using a single, relatively simple system, users today can often access a much larger and more diverse set of objects of interest than they could a few years ago—be they documents, products, or potential collaborators. It is therefore becoming relatively more attractive to delegate some of the work of dealing with these objects—even to a system which has an imperfect model of the user's requirements.

15.8.2 Increasing Feasibility of Successful Adaptation

As the need for user-adaptivity increases, so—fortunately—does its feasibility, largely because of advances in the following areas:

Ways of Acquiring Information About Users Most of the methods discussed in 15.5.2 for acquiring information about users are becoming more powerful with advances in technology and research. They therefore offer the prospect of substantial increases in the quality of adaptation—although methods for ensuring users' privacy call for equal attention.

Advances in Techniques for Learning, Inference, and Decision In addition to the more general progress in the fields of machine learning and artificial intelligence, communities of researchers have been focusing on the specific requirements of computational techniques that support user-adaptivity. Consequently, noticeable progress is being made every year in the areas discussed in Section 15.6.

Attention to Empirical Methods The special empirical issues and methods that are involved in the design and evaluation of user-adaptive systems have been receiving increasing attention of researchers, as emphasis has shifted from high technical sophistication to ensuring that the systems enhance the users' experience.

The future role of user-adaptive systems will not be the result of a sudden paradigm shift motivated by a desire to emulate interaction among humans. It will be shaped by continuing technical progress and increases in understanding along the many frontiers reviewed in this chapter.

15.9 Acknowledgements

Preparation of this chapter was supported by the German Ministry of Education, Research, Science, and Technology under grant 01 IW 001 as part of the MIAU project. Valuable comments on an earlier version were supplied by Eric Horvitz and by Rabia Aziz, Irfan Jaffry, Abdul Khan, Olga Lamonova, Shivaz Mehta, and Marie Norlien.

References

André, E., & Rist, T. (1995). Generating coherent presentations employing textual and visual material. *Artificial Intelligence Review, 9*, 147–165.

Balabanovic, M., & Shoham, Y. (1997). Fab: Content-based, collaborative recommendation. *Communications of the ACM, 40*(3), 66–72.

Billsus, D., & Pazzani, M. J. (2000). User modeling for adaptive news access. *User Modeling and User-Adapted Interaction, 10*, 147–180.

Bradley, K., Rafter, R., & Smyth, B. (2000). Case-based user profiling for content personalisation. In P. Brusilovsky, O. Stock, & C. Strapparava (Eds.), *Adaptive hypermedia and adaptive web-based systems: Proceedings of AH 2000* (pp. 62–72). Berlin: Springer.

Brusilovsky, P. (1996). Methods and techniques of adaptive hypermedia. *User Modeling and User-Adapted Interaction, 6*, 87–129.

Brusilovsky, P. (2001). Adaptive hypermedia. *User Modeling and User-Adapted Interaction, 11*, 87–110.

Budzik, J., Hammond, K., & Birnbaum, L. (2001). Information access in context. *Knowledge-Based Systems, 14*, 37–53.

Burke, R. D. (2000). Knowledge-based recommender systems. *Encyclopedia of Library and Information Science.*

Burke, R. D., Hammond, K. J., & Young, B. C. (1997). The FindMe approach to assisted browsing. *IEEE Expert, 12*(4), 32–40.

Carberry, S. (2001). Techniques for plan recognition. *User Modeling and User-Adapted Interaction, 11*, 31–48.

Chin, D. N. (2001). Empirical evaluation of user models and user-adapted systems. *User Modeling and User-Adapted Interaction, 11*, 181–194.

Collins, J. A., Greer, J. E., Kumar, V. S., McCalla, G. I., Meagher, P., & Tkatch, R. (1997). Inspectable user models for just-in-time workplace training. In A. Jameson, C. Paris, & C. Tasso (Eds.), *User modeling: Proceedings of the Sixth International Conference, UM97* (pp. 327–337). Vienna: Springer Wien New York.

Corbett, A. (2001). Cognitive computer tutors: Solving the two-sigma problem. In M. Bauer, P. Gmytrasiewicz, & J. Vassileva (Eds.), *UM2001, User modeling: Proceedings of the Eighth International Conference* (pp. 137–147). Berlin: Springer.

Corbett, A., McLaughlin, M., & Scarpinatto, K. C. (2000). Modeling student knowledge: Cognitive tutors in high school and college. *User Modeling and User-Adapted Interaction, 10*, 81–108.

Corbett, A. T., & Koedinger, K. R. (1997). Intelligent tutoring systems. In M. Helander, T. K. Landauer, & P. V. Prabhu (Eds.), *Handbook of human-computer interaction* (pp. 849–874). Amsterdam: North-Holland.

Dey, A. K., Abowd, G. D., & Wood, A. (1998). Cyberdesk: A framework for providing self-integrating context-aware services. *Knowledge-Based Systems, 11*, 3–13.

Fink, J., & Kobsa, A. (2000). A review and analysis of commercial user modeling servers for personalization on the world wide web. *User Modeling and User-Adapted Interaction, 10*, 209–249.

Fink, J., Kobsa, A., & Nill, A. (1998). Adaptable and adaptive information provision for all users, including disabled and elderly people. *New Review of Hypermedia and Multimedia, 4*, 163–188.

Ghosh, A. K., & Swaminatha, T. M. (2001). Software security and privacy risks in mobile e-commerce. *Communications of the ACM, 44*(2), 51–57.

Goecks, J., & Shavlik, J. (2000). Learning users' interests by unobtrusively observing their normal behavior. In H. Lieberman (Ed.), *IUI 2000: International Conference on Intelligent User Interfaces.* New York: ACM.

Good, N., Schafer, J. B., Konstan, J. A., Borchers, A., Sarwar, B., Herlocker, J., & Riedl, J. (1999). Combining collaborative filtering with personal agents for better recommendations. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence* (pp. 439–446). Orlando, FL.

Greer, J. E., McCalla, G. I., Collins, J. A., Kumar, V. S., Meagher, P., & Vassileva, J. (1998). Supporting peer help and collaboration in distributed workplace environments. *International Journal of AI and*

Education, *9*, 159–177.

Gutkauf, B., Thies, S., & Domik, G. (1997). A user-adaptive chart editing system based on user modeling and critiquing. In A. Jameson, C. Paris, & C. Tasso (Eds.), *User modeling: Proceedings of the Sixth International Conference, UM97* (pp. 159–170). Vienna: Springer Wien New York.

Haller, S., McRoy, S., & Kobsa, A. (Eds.). (1999). *Computational models of mixed-initiative interaction.* Dordrecht, The Netherlands: Kluwer.

Hammond, N. (1987). Principles from the psychology of skill acquisition. In M. M. Gardiner & B. Christie (Eds.), *Applying cognitive psychology to user-interface design.* Chichester, England: Wiley.

Han, J., & Kamber, M. (2001). *Data mining: Concepts and techniques.* San Francisco: Morgan Kaufmann.

Hanani, U., Shapira, B., & Shoval, P. (2001). Information filtering: Overview of issues, research and systems. *User Modeling and User-Adapted Interaction*, *11*, 203–259.

Heckerman, D. (1998). A tutorial on learning with Bayesian networks. In M. I. Jordan (Ed.), *Learning in graphical models.* Cambridge, MA: MIT Press.

Hegner, S. J., McKevitt, P., Norvig, P., & Wilensky, R. L. (Eds.). (2001). *Intelligent help systems for UNIX.* Dordrecht, the Netherlands: Kluwer.

Hirst, G., DiMarco, C., Hovy, E., & Parsons, K. (1997). Authoring and generating health-education documents that are tailored to the needs of the individual patient. In A. Jameson, C. Paris, & C. Tasso (Eds.), *User modeling: Proceedings of the Sixth International Conference, UM97* (pp. 107–118). Vienna: Springer Wien New York.

Horvitz, E. (1999). Principles of mixed-initiative user interfaces. In M. G. Williams, M. W. Altom, K. Ehrlich, & W. Newman (Eds.), *Human factors in computing systems: CHI '99 conference proceedings* (pp. 159–166). New York: ACM.

Horvitz, E., Breese, J., Heckerman, D., Hovel, D., & Rommelse, K. (1998). The Lumière project: Bayesian user modeling for inferring the goals and needs of software users. In G. F. Cooper & S. Moral (Eds.), *Uncertainty in Artificial Intelligence: Proceedings of the Fourteenth Conference* (pp. 256–265). San Francisco: Morgan Kaufmann.

Horvitz, E., Jacobs, A., & Hovel, D. (1999). Attention-sensitive alerting. In K. B. Laskey & H. Prade (Eds.), *Uncertainty in Artificial Intelligence: Proceedings of the Fifteenth Conference* (pp. 305–313). San Francisco: Morgan Kaufmann.

Höök, K. (2000). Steps to take before IUIs become real. *Journal of Interaction with Computers.*

Jameson, A. (1996). Numerical uncertainty management in user and student modeling: An overview of systems and issues. *User Modeling and User-Adapted Interaction*, *5*, 193–251.

Jameson, A., Großmann-Hutter, B., March, L., Rummer, R., Bohnenberger, T., & Wittig, F. (2001). When actions have consequences: Empirically based decision making for intelligent user interfaces. *Knowledge-Based Systems*, *14*, 75–92.

Jensen, F. V. (2001). *Bayesian networks and decision graphs.* New York: Springer.

Joachims, T., Freitag, D., & Mitchell, T. (1997). WebWatcher: A tour guide for the World Wide Web. In M. E. Pollack (Ed.), *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence* (pp. 770–777). San Francisco, CA: Morgan Kaufmann.

Jones, R., Pearson, J., McGregor, S., Cawsey, A. J., Barrett, A., Craig, N., Atkinson, J. M., Gilmour, W. H., & McEwen, J. (1999). Randomised trial of personalised computer based information for cancer patients. *British Medical Journal*, *319*, 1241–1247.

King, M. F., & Bruner, G. C. (2000). Social desirability bias: A neglected aspect of validity testing. *Psychology & Marketing*, *17*, 79–103.

Kobsa, A. (2001). Generic user modeling systems. *User Modeling and User-Adapted Interaction*, *11*, 49–63.

Kobsa, A., & Wahlster, W. (Eds.). (1989). *User models in dialog systems.* Berlin: Springer.

Krulwich, B. (1997). Lifestyle Finder: Intelligent user profiling using large-scale demographic data. *AI Magazine*, *18*(2), 37–45.

Lajoie, S. P., & Vivet, M. (Eds.). (1999). *Artificial intelligence in education: Open learning environments: New computational technologies to support learning, exploration, and collaboration.* Amsterdam: IOI Press.

Langley, P. (1996). *Elements of machine learning.* San Francisco: Morgan Kaufmann.

Langley, P., & Fehling, M. (1998). *The experimental study of adaptive user interfaces* (Tech. Rep. No. Technical Report 98-3). Institute for the Study of Learning and Expertise, Palo Alto, CA.

Lanier, J. (1995). Agents of alienation. *interactions, 2*(3), 66–72.

Lieberman, H. (Ed.). (2001). *Your wish is my command: Programming by example.* San Francisco: Morgan Kaufmann.

Linden, G., Hanks, S., & Lesh, N. (1997). Interactive assessment of user preference models: The automated travel assistant. In A. Jameson, C. Paris, & C. Tasso (Eds.), *User modeling: Proceedings of the Sixth International Conference, UM97* (pp. 67–78). Vienna: Springer Wien New York.

Litman, D. J., & Pan, S. (2000). Predicting and adapting to poor speech recognition in a spoken dialogue system. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence* (pp. 722–728). Austin, TX.

Maes, P. (1994). Agents that reduce work and information overload. *Communications of the ACM, 37*(7), 30–40.

Maglio, P. P., Barrett, R., Campbell, C. S., & Selker, T. (2000). SUITOR: An attentive information system. In H. Lieberman (Ed.), *IUI 2000: International Conference on Intelligent User Interfaces* (pp. 169–176). New York: ACM.

Maglio, P. P., & Campbell, C. S. (2000). Tradeoffs in displaying peripheral information. In T. Turner, G. Szwillus, M. Czerwinski, & F. Paternò (Eds.), *Human factors in computing systems: CHI 2000 conference proceedings* (pp. 241–248). New York: ACM.

Martin, D. M., Smith, R. M., Brittain, M., Fetch, I., & Wu, H. (2001). The privacy practices of web browser extensions. *Communications of the ACM, 44*(2), 45–50.

McDonald, D. W., & Ackerman, M. S. (2000). Expertise Recommender: A flexible recommendation system and architecture. In *Proceedings of the 2000 Conference on Computer-Supported Cooperative Work* (pp. 231–240).

Mitchell, T., Caruana, R., Freitag, D., McDermott, J., & Zabowski, D. (1994). Experience with a learning personal assistant. *Communications of the ACM, 37*(7), 81–91.

Mitchell, T. M. (1997). *Machine learning.* Boston: McGraw-Hill.

Mladenic, D. (1999). Text-learning and related intelligent agents: A survey. *IEEE Intelligent Systems, 14*(4), 44–54.

Müller, C., Großmann-Hutter, B., Jameson, A., Rummer, R., & Wittig, F. (2001). Recognizing time pressure and cognitive load on the basis of speech: An experimental study. In M. Bauer, P. Gmytrasiewicz, & J. Vassileva (Eds.), *UM2001, User modeling: Proceedings of the Eighth International Conference.* Berlin: Springer.

Norman, D. A. (1994). How might people interact with agents. *Communications of the ACM, 37*(7), 68–71.

Paiva, A. (1997). Learner modelling for collaborative learning environments. In B. du Boulay & R. Mizoguchi (Eds.), *Artificial intelligence in education: Knowledge and media in learning systems* (pp. 215–222). Amsterdam: IOI Press.

Pazzani, M., & Billsus, D. (1997). Learning and revising user profiles: The identification of interesting web sites. *Machine Learning, 27*, 313–331.

Pazzani, M. J. (1999). A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review, 13*, 393–408.

Pazzani, M. J., & Billsus, D. (1999). Evaluating adaptive web site agents. In *Proceedings of the Workshop on Recommender Systems Algorithms and Evaluation, 22nd International Conference on Research and Development in Information Retrieval.*

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference.* San Mateo, CA: Morgan Kaufmann.

Picard, R. W. (1997). *Affective computing.* Cambridge, MA: MIT Press.

Rhodes, B. J. (2000). *Just-in-time information retrieval.* Unpublished doctoral dissertation, School of Architecture and Planning, Massachusetts Institute of Technology, Cambridge, MA. (Available from http://rhodes.www.media.mit.edu/people/rhodes/research/)

Rich, E. (1979). User modeling via stereotypes. *Cognitive Science, 3,* 329–354.

Rich, E. (1989). Stereotypes and user modeling. In A. Kobsa & W. Wahlster (Eds.), *User models in dialog systems* (pp. 35–51). Berlin: Springer.

Rössel, M. (2000). *Ein System zur individualisierten Informationsvermittlung – dargestellt am Beispiel eines multimedialen Branchenkatalogs der Technischen Keramik [A system for individualized information presentation—as exemplified by a multimedia product catalog for technical ceramics].* Unpublished doctoral dissertation, Bereich Wirtschaftsinformatik I, Universität Erlangen-Nürnberg.

Sawhney, N., & Schmandt, C. (2000). Nomadic Radio: Speech and audio interaction for contextual messaging in nomadic environments. *ACM Transactions on Computer-Human Interaction, 7,* 353–383.

Schafer, J. B., Konstan, J., & Riedl, J. (1999). Recommender systems in e-commerce. In *Proceedings of the ACM Conference on Electronic Commerce.*

Schaumburg, H. (2001). Computers as tools or as social actors? - The users' perspective on anthropomorphic agents. *International Journal on Intelligent Cooperative Information Systems, 10.*

Schiele, B., Starner, T., Rhodes, B., Clarkson, B., & Pentland, A. (2000). Situation aware computing with wearable computers. In W. Barfield & T. Caudell (Eds.), *Augmented reality and wearable computers.* Mahwah, NJ: Erlbaum.

Sears, A., & Shneiderman, B. (1994). Split menus: Effectively using selection frequency to organize menus. *ACM Transactions on Computer-Human Interaction, 1,* 27–51.

Segal, R. B., & Kephart, J. O. (1999). MailCat: An intelligent assistant for organizing e-mail. In *Proceedings of the Third International Conference on Autonomous Agents* (pp. 276–282).

Segal, R. B., & Kephart, J. O. (2000). Incremental learning in SwiftFile. In P. Langley (Ed.), *Machine Learning: Proceedings of the 2000 International Conference.* San Francisco: Morgan Kaufmann.

Shearin, S., & Lieberman, H. (2001). Intelligent profiling by example. In J. Lester (Ed.), *IUI 2001: International Conference on Intelligent User Interfaces* (pp. 145–151). New York: ACM.

Shneiderman, B. (1995). Looking for the bright side of user interface agents. *interactions, 2*(1), 13–15.

Shneiderman, B., & Maes, P. (1997). Direct manipulation vs. interface agents. *interactions, 4*(6), 42–61.

Trewin, S., & Pain, H. (1998). A model of keyboard configuration requirements. In *Proceedings of the Third International ACM Conference on Assistive Technologies* (pp. 173–181).

Vivacqua, A., & Lieberman, H. (2000). Agents to assist in finding help. In T. Turner, G. Szwillus, M. Czerwinski, & F. Paternò (Eds.), *Human factors in computing systems: CHI 2000 conference proceedings* (pp. 65–72). New York: ACM.

Wainer, H. (Ed.). (2000). *Computerized adaptive testing: A primer* (2nd ed.). Hillsdale, NJ: Erlbaum.

Webb, G., Pazzani, M. J., & Billsus, D. (2001). Machine learning for user modeling. *User Modeling and User-Adapted Interaction, 11,* 19–29.

Weber, G., & Specht, M. (1997). User modeling and adaptive navigation support in WWW-based tutoring systems. In A. Jameson, C. Paris, & C. Tasso (Eds.), *User modeling: Proceedings of the Sixth International Conference, UM97* (pp. 289–300). Vienna: Springer Wien New York.

Wexelblat, A., & Maes, P. (2001). *Issues for software agent UI.* (Unpublished manuscript, available from http://wex.www.media.mit.edu/people/wex/)

Wittig, F., & Jameson, A. (2000). Exploiting qualitative knowledge in the learning of conditional probabilities of Bayesian networks. In C. Boutilier & M. Goldszmidt (Eds.), *Uncertainty in Artificial Intelligence: Proceedings of the Sixteenth Conference* (pp. 644–652). San Francisco: Morgan Kaufmann.

Wolfman, S. A., Lau, T., Domingos, P., & Weld, D. S. (2001). Mixed initiative interfaces for learning tasks: SMARTedit talks back. In J. Lester (Ed.), *IUI 2001: International Conference on Intelligent User Interfaces* (pp. 167–174). New York: ACM.

Zukerman, I., & Albrecht, D. W. (2001). Predictive statistical models for user modeling. *User Modeling and User-Adapted Interaction*, *11*, 5–18.

Zukerman, I., & Litman, D. (2001). Natural language processing and user modeling: Synergies and limitations. *User Modeling and User-Adapted Interaction*, *11*, 129–158.

Incremental Learning in SwiftFile

Richard B. Segal RSEGAL@WATSON.IBM.COM
Jeffrey O. Kephart KEPHART@WATSON.IBM.COM
IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598

Abstract

SwiftFile is an intelligent assistant that helps users organize their e-mail into folders. SwiftFile uses a text classifier to predict where each new message is likely to be filed by the user and provides shortcut buttons to quickly file messages into one of its predicted folders. One of the challenges faced by SwiftFile is that the user's mail-filing habits are constantly changing — users are frequently creating, deleting and rearranging folders to meet their current filing needs. In this paper, we discuss the importance of incremental learning in SwiftFile. We present several criteria for judging how well incremental learning algorithms adapt to quickly changing data and evaluate SwiftFile's classifier using these criteria. We find that SwiftFile's classifier is surprisingly responsive and does not require the extensive training that is often assumed in most learning systems.

1. Introduction

SwiftFile is an intelligent assistant that helps users organize their e-mail into folders (Segal & Kephart, 1999). Using a text classifier that dynamically adjusts to the user's mail-filing habits, SwiftFile predicts for each incoming message the three folders that it deems most likely to be chosen by the user as destinations. The user may then file a message into one of these predicted folders by clicking on a shortcut button representing the desired folder. If none of SwiftFile's predictions are correct, the user can simply resort to the usual method for filing messages.

Previously, SwiftFile's classification accuracy was assessed using standard static evaluation techniques. However, static evaluation is not entirely satisfactory. The e-mail environment is highly dynamic, with folders and messages constantly being created, destroyed, and reorganized. A classifier that is useful today may be much less so a month from now.

Static analysis fails to address several key questions about how well a dynamic learner like SwiftFile is likely to perform in a dynamic environment.

1. How quickly can SwiftFile adapt to new users?

   SwiftFile can bootstrap itself by learning from a user's previously-filed messages when it is first installed. However, if the user is new to e-mail and does not have any previously-filed messages, there is no data available to initialize the classifier. How many messages does the user have to file before SwiftFile can make useful suggestions?

2. How well does SwiftFile track the changing environment of established users?

   An important and frequent source of change is the creation of a new folder. A new folder may reflect either a change in the stream of mail received by a user or a change in that user's filing habits. Initially, new folders contain only a handful of messages. How many messages will SwiftFile have to see in a new folder before it can start suggesting which messages should be placed there?

3. How important is incremental learning?

   Some systems designed to help users prioritize or organize their e-mail do not use incremental learning (Maes, 1994; Payne & Edwards, 1997). The authors of these systems address the need for adaptation by recommending retraining the system from scratch, perhaps on a daily basis. Is retraining a classifier on a daily basis sufficient in a highly-dynamic environment like e-mail?

After a brief description of SwiftFile and a brief review of results obtained from a standard static evaluation, we develop a variety of dynamic evaluation measures inspired by these questions and use them to assess the performance of SwiftFile's incremental learning algo-

rithm. We then conclude with a brief discussion, including a comparison to related work.

2. SwiftFile

SwiftFile is an add-on to Lotus Notes that helps users file their e-mail into folders. Figure 1 shows SwiftFile in action. SwiftFile places three *MoveToFolder* shortcut buttons above each message. The shortcut buttons allow the user to quickly move the message into one of the three folders that SwiftFile predicts to be the message's most-likely destinations. The buttons are ordered from top to bottom, with the topmost representing SwiftFile's best guess and the bottommost representing its third-best guess. When one of the three buttons is clicked, the message is immediately moved to the indicated folder.

SwiftFile's predictions are made using a text classifier. Classifiers often require a large number of training messages before they yield accurate predictions, but SwiftFile circumvents this potential problem. During installation, it treats previously-filed messages as a corpus of labeled documents and uses them to train its text classifier. After its initial training, SwiftFile is immediately ready to make accurate predictions.

SwiftFile adapts to changing conditions by using incremental learning. Once the classifier has been trained, the classifier's model is continuously updated by presenting to the classifier the messages that have been added to or deleted from each folder. The cost of the update is only linear in the length of the message. After updating, the classifier's predictions are identical to those that would be obtained by training the classifier from scratch on the entire mail database.

3. AIM

SwiftFile uses a modified version of AIM to classify messages (Barrett & Selker, 1995). AIM is a TF-IDF style text classifier. We have modified the original AIM implementation to support incremental learning.

AIM represents each message $\mathcal{M}$ as a word-frequency vector $F(\mathcal{M})$, in which each component $F(\mathcal{M}, w)$ represents the total number of times the word w appears in $\mathcal{M}$. Each folder $\mathcal{F}$ is represented using a *weighted* word-frequency vector $W(\mathcal{F}, w)$. Several steps are involved in computing $W(\mathcal{F}, w)$. First, the folder $\mathcal{F}$'s centroid vector $F(\mathcal{F}, w)$ is computed by summing the word-frequency vectors for each message in the folder:

$$F(\mathcal{F}, w) = \sum_{\mathcal{M} \in \mathcal{F}} F(\mathcal{M}, w). \tag{1}$$

This folder centroid vector is then converted to a weighted word-frequency vector using the TF-IDF principle: the weight assigned to a word is proportional to its frequency in the folder and inversely proportional to its frequency in other folders. We define $FF(\mathcal{F}, w)$ to be the fractional frequency of word w among messages contained within folder $\mathcal{F}$, or the number of times word w occurs in folder $\mathcal{F}$ divided by the total number of words in $\mathcal{F}$:

$$FF(\mathcal{F}, w) = \frac{F(\mathcal{F}, w)}{\sum_{w' \in \mathcal{F}} F(\mathcal{F}, w')}. \tag{2}$$

The definition of term frequency $TF(\mathcal{F}, w)$ used by AIM is

$$TF(\mathcal{F}, w) = FF(\mathcal{F}, w) / FF(\mathcal{A}, w), \tag{3}$$

where $\mathcal{A}$ represents the set of all messages (the entire database of organized mail). We define the document frequency $DF(w)$ to be the fraction of folders in which the word w appears at least once. The definition of inverse document frequency $IDF(w)$ used by AIM is

$$IDF(w) = \frac{1}{DF(w)^2}. \tag{4}$$

Finally, AIM combines these two formulas to define the weight for word w in folder $\mathcal{F}$:

$$W(\mathcal{F}, w) = TF(\mathcal{F}, w) \times IDF(w). \tag{5}$$

The weight vectors for each folder are used to classify each new message. When a message $\mathcal{M}$ arrives to be classified, it is first converted into a word-frequency vector $F(\mathcal{M})$. Then, AIM computes the similarity between $\mathcal{M}$ and the weighted word-frequency vectors for each folder, $W(\mathcal{F})$. AIM computes the similarity between the message vector and the weighted folder vectors using a variation of cosine distance called *SIM4* (Salton & McGill, 1983):

$$SIM4(\mathcal{M}, \mathcal{F}) = \frac{\sum_{w \in \mathcal{M}} F(\mathcal{M}, w)\, W(\mathcal{F}, w)}{\min(\sum_{w \in \mathcal{M}} F(\mathcal{M}, w), \sum_{w \in \mathcal{M}} W(\mathcal{F}, w))}. \tag{6}$$

Here the sums are taken only over the words that are contained within $\mathcal{M}$. Finally, AIM takes the three folders with greatest similarity as its predictions.

The original AIM classifier implemented this classification technique as a batch algorithm. We modified AIM's implementation to support incremental